TOWARDS AUTONOMOUS MANAGEMENT OF LARGE-SCALE WIRELESS SENSOR NETWORKS UTILIZING MULTI-PARENT RECURSIVE AREA HIERARCHIES

By

JOHNATHAN VEE CREE

A dissertation submitted in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

MAY 2013

To the Faculty of Washington State University:

The members of the Committee appointed to examine the dissertation of JOHNATHAN VEE CREE find it satisfactory and recommend that it be accepted.

Jose Delgado-Frias, Ph.D., Chair

David Bakken, Ph.D.

Carl Hauser, Ph.D.

ACKNOWLEDGMENTS

I would like to thank Jose Delgado-Frias for his support and assistance as an adviser and providing me with various opportunities throughout my graduate career. Michael Hughes, Bruce Lawler, Ivan Amaya and Laura Cree for their technical assistance and discussions of many of the concepts and ideas that resulted in the presented solutions. Thanks to Brion Burghard and Kurt Silvers for their time, support and help. Thank you to Laura Cree, Bruce Lawler, and Shannon Crowell for their time and help with reviewing and editing.

TOWARDS AUTONOMOUS MANAGEMENT OF LARGE-SCALE WIRELESS SENSOR NETWORKS UTILIZING MULTI-PARENT RECURSIVE AREA HIERARCHIES

Abstract

by Johnathan Vee Cree, Ph.D.
Washington State University
MAY 2013

Chair: Jose Delgado-Frias

Large scale, duty-cycled, wireless sensor networks distributed across a large geographic area have been proposed for applications ranging from anomaly detection to vehicle tracking. In order to support the requirements of these applications an efficient and effective network management method that autonomously configures and maintains the network is required. When selecting a network management method it is important to consider both the direct and indirect costs associated with the different solution. An indirect cost that can be significant in duty-cycled networks is the overhead associated with local synchronization for communication. Further, an effective solution needs to recognize that in-network data aggregation and analysis presents significant benefits to wireless sensor networks and should configure the network to provide inherent benefits when said higher level functions utilize the structure as a paradigm. NOA, the proposed network management protocol, provides a multi-

parent hierarchical logical structure for a network. NOA utilizes the multi-parent structure to reduce the cost of local synchronization as well as hierarchy creation and maintenance. A set of models has been designed and presented in order to compare the construction and data aggregation cost of NOA with other protocols. The ns-3 simulator was extended to provide a simulation environment for NOA and hierarchical beaconing and simulation data is presented to further verify the reduced management cost. The simulator also confirms that the reduced management costs are significant enough to extend the lifetime of a network configured with NOA compared to a network configured using hierarchical beaconing. The multi-parent structure provided by NOA can also provide higher level functions with benefits such as, but not limited to: removing network divisions that are encountered in single-parent hierarchies, data comparison between a device and its neighbors at a common grandparent, and redundancies for communication paths as well as in-network data aggregation, analysis and storage.

TABLE OF CONTENTS

ACKNOWLEDGMENTS iii

ABSTRACT iv

LIST OF TABLES viii

LIST OF FIGURES ix

1. Introduction 1

    1.1 Background 4
    1.2 Related Work 22
    1.3 Naming Conventions 24
    1.4 Overview 25

2. Network Model 26

    2.1 Configuration Algorithms and Ideal Distribution 26
    2.2 One-, Two, and Three-Parent Hierarchies 27
    2.3 Hierarchy Construction Cost Model 36
    2.4 Aggregation Cost Model 45
    2.5 Synchronization 47

3. NOA: Multi-Parent Recursive Area Hierarchy Management Protocol 49

    3.1 NOA: Multi-Parent Hierarchy Construction 50
    3.2 Cluster-head Selection with Location Aware Devices 52
    3.3 NOA: Multi-Parent Hierarchy Maintenance 55
    3.4 NOA_{N-CH} and NOA_{1-CH} Variants 56
    3.5 Cluster Synchronization 58

4. Simulation Environment 60

4.1 Simulator Design 60
4.2 A Simulation Run 67
4.3 Simulations 69

5. Evaluation and Analysis 72

5.1 Discussion of Single- and Multi-Parent Hierarchies 72
5.2 Hierarchy Construction 76
5.3 Data Aggregation 84
5.4 Network Lifetime 86
5.5 Summary of Results 90

6. Conclusion 95

6.1 Contributions 95
6.2 Future Work 99

LIST OF TABLES

Table Page

5.1 Simulated Construction Cost of N-CH Hierarchies 77

5.2 Round Robin Data Aggregation Cost per Round for N-CH Hierarchies 84

5.3 Number of Tiers vs. Number of Nodes and Hierarchy Type 87

LIST OF FIGURES

Figure		Page
1.1	Conceptual Single-Parent Hierarchy	3
1.2	Conceptual Multi-Parent Hierarchy	4
2.1	Network Model Ideal Distribution	28
2.2	One-, Two-, and Three-Parent Example Hierarchies	29
2.3	Tiers vs. Number of Nodes	37
2.4	Radius of a Cluster	38
2.5	Percent of Nodes Acting as Cluster-heads	39
2.6	Total Number of Cluster-heads	40
3.1	Hierarchy Creation Progression	51
3.2	1-CH vs. N-CH	59
4.1	One-Parent Hierarchy Random Distribution	64
4.2	Two-Parent Hierarchy Random Distribution	65
5.1	Simulated Construction Cost of N-CH Hierarchies	77
5.2	Construction Cost N-CH vs. 1-CH	79
5.3	Construction Cost Simulation vs. Model	81
5.4	Aggregation Cost Simulation vs. Model	85
5.5	Network Lifetime vs. Network Size	88
5.6	Network Lifetime vs. Synchronization, Data Aggregation and Maintenance Period	89
5.7	Network Lifetime vs. Synchronization Period	91

5.8 Network Lifetime vs. Data Aggregation and Maintenance Period 92

Dedication

Dedicated to: my loving and supportive wife who encourages me to make every day an adventure even when she gets stung, muddy and tired; my parents for raising me to believe that I could be and do anything that I imagined and for instilling in me a sense of creativity, motivation, and purpose; my brother for providing me with philosophical arguments that required critical analysis from our childhood until the end of ...?; my grandmother for providing me a work ethic and my granddad for always listening to my ideas...

CHAPTER 1. INTRODUCTION

Large-scale wireless sensor networks (WSNs) require the collaboration of thousands to millions of sensor devices spread across a vast geographic area. Applications of large-scale WSNs include, but are not limited to environment/habitat monitoring, area intrusion detection, and person/vehicle tracking Bhatti and Xu [2009], Xie et al. [2011], Alippi et al. [2011], Tseng et al. [2003], Oliveira and Rodrigues [2011], Dyo et al. [2012], Naumowicz et al. [2010]. When dealing with a large number of devices, autonomous setup and configuration significantly reduces the amount of human effort required to install a network. Autonomous configuration lowers the cost of applying a WSN solution and reduces errors during installation. It also allows the configuration to dynamically adapt to changes in the network and environment.

While autonomous configuration provides many benefits, it is not free. The costs associated with configuring and maintaining an autonomously configured network of resource constrained WSNs, such as those with battery powered devices, is of great concern as this cost can have a significant impact on a network's lifetime. The constrained resources of a wireless sensor device go beyond that of the power reserves and also include the limited computational power and communication bandwidth. As such, it is necessary to utilize a scalable solution to autonomously organize

sensor devices that promotes radio duty cycling, efficient and effective routing, and an architecture for higher level functions (e.g. in-network data analysis, distributed hash tables, query resolution, distributed content broker etc. Xie et al. [2011], Jha and Sharma [2011], Andreou et al. [2011], Rajagopalan and Varshney [2006], il Hwang et al. [2006], Kimura and Latifi [2005], Jardak et al. [2007], Aly et al. [2011], Quiroz and Parashar [2006], Fasolo et al. [2007]) to extend the lifetime of a wireless sensor network.

A recursive area hierarchy has been shown to be a promising scalable solution for autonomously organizing wireless sensor networks Iwanicki and van Steen [2010]. The recursive area hierarchy is a logical overlay on the network. It organizes the devices into a hierarchy of clusters starting with $tier^1$ clusters and recursively clustering the lower tiered clusters into a super-cluster until the top most cluster covers the entire network. In other words, $tier^1$ clusters consist of a group of devices in a given proximity, $tier^2$ clusters are composed of $tier^1$ clusters in an area exponentially larger than the $tier^1$ clusters. $Tier^3$ clusters are composed of $tier^2$ clusters and so on until the highest level $tier^N$ cluster. This type of structural overlay provides a scalable solution that is logarithmic in terms of the number of tiers in the hierarchy and can be used as a paradigm for higher level functions such as routing, multi-resolution analysis and multi-resolution data storage.

Current state-of-the-art solutions for recursive area hierarchies focus on a single-

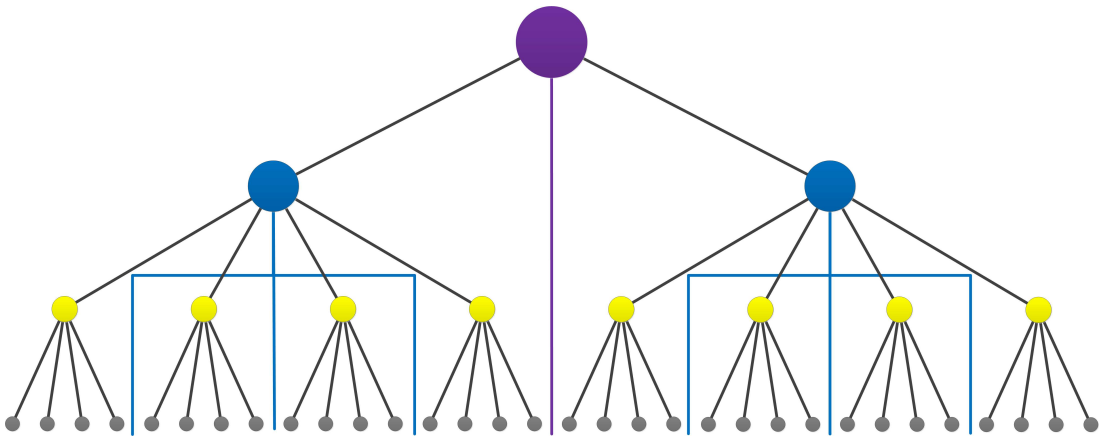


Figure 1.1: A conceptual representation of a single-parent hierarchy. The single-parent clustering creates a single point of failure at each cluster-head and divides the network into separate groups that cannot share information among neighboring clusters without traversing the hierarchy and communicating through the common ancestor. This makes it more difficult for a cluster to compare its information with all of its neighboring clusters as it only has direct access to those within its super-cluster.

parent hierarchy (Fig. 1.1) where each cluster is only part of a single super-cluster. The single parent structure divides the network at each cluster into disjoint sets. These divisions can be viewed as taking a mesh network and creating a hierarchy of logical star-networks on top of it. This can artificially limit the capabilities of the mesh network by reducing the connectivity of the devices. The multi-parent hierarchy described herein (Fig. 1.2) removes the artificial limitations introduced by the single-parent hierarchies while creating a scalable structure that can be constructed, maintained and utilized at a comparable cost.

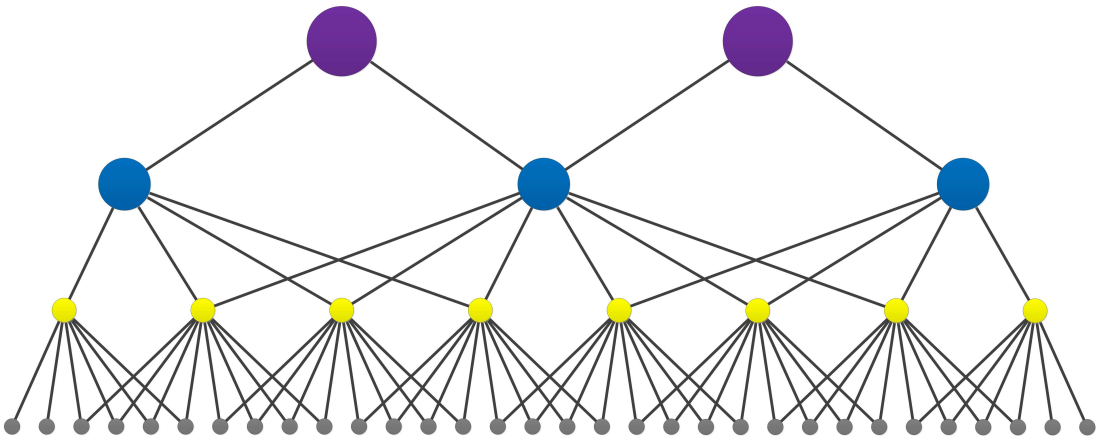


Figure 1.2: A conceptual representation of a two-parent hierarchy. Since each device can have multiple parents the mesh-network is not artificially divided into a hierarchy of star networks and redundancy can be easily built into the hierarchy. This allows for information from neighboring clusters to be compared at the common parents making it possible to compare a cluster with all of its neighbors instead of only those neighbors that are in its cluster.

1.1 Background

The lifetime of a wireless sensor network is impacted by the individual devices, communication protocols, and the logical structure of the network. These are closely coupled entities that significantly impact each other. For example, the power consumption of a device can be significantly reduced by duty-cycling a device's radio, sensors and/or the entire device. However, duty-cycling a device's radio has significant impacts on communication, as neighboring devices must be on in order to communicate which requires some form of synchronization. The logical structure of a network and the protocols used to generate that structure can require different

levels of synchronization based on the routing mechanisms used (i.e. broadcasting, flooding, uni-cast, multi-cast). These are just a few of the many examples of coupling between the device, its communication protocols and the logical structure of the network that need to be considered when designing an autonomous management protocol for duty-cycled large-scale WSNs.

This section focuses on the areas of research that most significantly impacted the final design of the proposed protocol. The discussion starts with clustering §1.1.1 and how clustering can reduce the number of devices that need to interact for local decisions and network wide decisions. The recursive area clustering hierarchy discussed in this section introduces a varying scope of resolution that is similar to the varying degrees of governments (i.e. city, county, state, national, global). Sub-section §1.1.2 provides a use-case for spatiotemporal multi-resolution and discusses how a recursive area clustering hierarchy can be used to reduce the communication between a device and the higher levels of analysis while retaining useful information in the network for future analysis. The last sub-section §1.1.3 provides a brief background on local synchronization for duty-cycling devices including synchronization periods, mechanisms and different classifications of synchronization.

1.1.1 Clustering

Clustering is the process of grouping nodes that are within a given proximity of each other. A cluster-head is a node that takes on the task of managing a cluster Boyinbode et al. [2010], Abbasi and Younis [2007]. In certain situations a cluster-head can take on application specific tasks such as data aggregation in order to meet cluster and/or network goals. The complexity of performing localized optimization tasks can be reduced when clustering because there are typically fewer devices which cover a smaller area and there is a device used to manage each cluster (i.e. the cluster-head). Clustering can also reduce the complexity of network wide optimization tasks as there are typically fewer cluster-heads than devices in the network. The reduction in the complexity of optimization makes clustering an ideal paradigm to perform network optimization tasks such as: scheduling, load sharing, power balancing, in-network analysis, sensor/communication coverage when duty-cycling, etc.

Over time clustering protocols have evolved to address the various requirements of proposed sensor network applications. The first clustering techniques involved one level of clustering to reduce the number of devices communicating with a data sink. As more devices were added to a network, new approaches for one-hop and multi-hop WSNs were designed to scale with the systems. One-hop solutions assume that every device in the network can communicate with every other device in the network

whereas multi-hop approaches deal with networks spread across vast areas and can require multi-hop communication between devices. This section primarily focuses on how solutions evolved to meet the requirements of different systems with regard to network size and spread.

1.1.1.1 Flat Clustering

Flat clustering techniques break the network up into a set of clusters where each cluster-head directly reports to a data sink. This type of solution reduces the number of devices that communicate with the data sink and allows the cluster-head to aggregate data from children before forwarding it to the data-sink. Solutions such as the LEACH protocol Handy et al. [2007] require one hop communication between a cluster-head and the data sink. MR-LEACH Farooq et al. [2010], however, addresses this limitation and allows multi-hop communication between cluster-heads and the data sink.

Flat clustering is not scalable because as the number of devices in a network increases so does the number of cluster-heads that report to the data sink. As a result, when there is a significant increase in the number of cluster-heads, network congestion as well as data implosion issues can occur.

1.1.1.2 Recursive Area Clustering Hierarchies

Recursive area clustering hierarchies allow for clusters to form super-clusters, (i.e. clusters of clusters, and super-clusters to be clustered and so on until every device is part of the highest tiered cluster). This provides a scalable solution because the number of tiers required for a recursive area hierarchy is O(log(n)), where n is the number of nodes in a network. Organization of a network into a recursive area hierarchy is a difficult problem, and optimizing the node states (routing information, parent-child connections, etc.) of a recursive area hierarchy is an NP-complete problem Hagouel [1983] as such a heuristic is required for an efficient solution. Further, when dealing with WSNs a distributed clustering mechanism is necessary as the cost of sending the network topology to a central location and then distributing the hierarchy structure to the network would be extremely high.

Current distributed approaches for autonomous configuration of recursive area hierarchies utilize either a top-down or bottom-up methodology. Top-down solutions start at the top and split the area up into successively smaller clusters. These algorithms make the assumption that the network is densely distributed where every device can communicate directly with any other device in the network (also known as single-hop hierarchies). Bottom-up protocols allow for sensor networks that are distributed across a vast area that can require multiple hops between any two devices in the network. Due to the inefficiencies with building multi-hop recursive area

hierarchies using a top-down method these solutions start at the bottom and group smaller clusters into larger super-clusters until the top-most cluster(s) cover the entire network.

Top-down protocols for recursive area hierarchies, Okeke and Law [2008], Jin et al. [2008], Nazir and Hasbullah [2010], Soni and Chand [2010], Mamuny et al. [2011], assume that all nodes can communicate directly with each other if the transmission power is dynamically increased. Clustering is used to allow each device to minimize its radio's transmission power to a level where it can communicate with its cluster-head. As such the hierarchies can reduce the power consumption when performing data collection. Since the transmission power does not scale linearly with the distance of a transmission, significant power savings have been possible with single-hop recursive area hierarchies. The single-hop hierarchy also helps to relieve some of the congestion seen in a single level clustering scheme. This is because a device with a lower transmission power will affect a smaller area when transmitting, allowing devices that are not in the same transmission area to simultaneously send data without interference. The assumption that a large-scale WSN is densely distributed in a relatively small area, however, is not practical Iwanicki and Van Steen [2009]. Many of the real world applications for large-scale WSNs deal with devices spread across vast areas and require multi-hop communications.

Current state-of-the-art multi-hop recursive area hierarchy protocols utilize a

bottom-up construction method and propagate local hierarchy information to all nodes in a cluster either by flooding (e.g. hierarchical beaconing (HB) Du et al. [2008, 2004], PalChaudhuri et al. [2005], Bandyopadhyay and Coyle [2003]) or by single-hop broadcasts where the hierarchy information is merged with its own heartbeat on receipt, as is the case with the gossip-based (GB) solutions Iwanicki and Van Steen [2009], Iwanicki and van Steen [2010]. Either solution requires that the information about every cluster-head be propagated to all of the devices in the cluster and uses a broadcast type of transmission to do this. When dealing with a sparse network, propagation by broadcast can require that all devices be synchronized with all of their neighbors. This type of synchronization is an inefficient use of resources for many WSN applications that do not explicitly require broadcasting.

1.1.1.3 Single-Parent Recursive Area Clustering Hierarchies

One way to think of the single-parent hierarchy (Fig. 1.1) is to break the sensor network into a set of distinct single-hop star networks, then build a hierarchy of logical star networks by using the smaller star networks to create a set of larger distinct star networks and so on. This solution, however, creates network divisions at the edge of each cluster (star) where a device in one cluster cannot communicate with a device in a neighboring cluster without violating the hierarchical structure. Network divisions create artificial communication limitations that can create issues in sensor networks involving devices sparsely dispersed across a vast region.

In a sparse network the single-parent hierarchy structure causes a device/cluster to be compared with only the neighboring devices/clusters that are part of the same cluster. This significantly reduces the amount of local information that is used to detect anomalies which can create errors, especially when these devices are distributed where there is a significant physical change in the environment. For example, if a device captures a phenomenon that the neighbors it is compared with did not capture, but another neighboring device did, the phenomenon could be incorrectly recorded. Depending on the configuration the phenomenon might be overlooked as a glitch or it could be incorrectly reported as an anomaly. Until the data is further analyzed at the common ancestor there is no way to determine which case is correct, without violating the hierarchical structure. In the case where the solution allows for the application to disregard the hierarchical structure, communication bridges can be used between clusters to allow neighboring clusters to communicate. The communication bridge type solution increases the complexity of data analysis routines because said routines need to determine if they should invoke the expensive process of multi-hop communication in order to determine if the information at a neighbor is useful.

1.1.1.4 Multi-Parent Hierarchies

The multi-parent hierarchy (Fig. 1.2) is equally as scalable as the single-parent hierarchy Cree et al. [2012], Cree [2012]. This structure maintains the simplicity of

maintenance with the star-network hierarchy but creates a mesh component by allowing a device/cluster-head to join multiple star networks, thus linking neighboring star networks at each level through that device. This allows the data from a device/cluster-head to be sent to all of the cluster-heads above it where it will be compared to the neighbors that are part of that cluster. The data from a device/cluster and all of its neighbors is not necessarily compared with all of its neighbors at a single parent, but due to the multi-parent structure the data from a device/cluster-head and all of its neighbors will be compared at a common grand-parent (i.e. two tiers above the current cluster). This property can be used to reduce the complexity of higher level functions because now these functions do not have to determine when/if they need to route data to a neighboring cluster for comparison.

Care must be taken when performing data aggregation in multi-parent hierarchies. A mechanism needs to be in place to ensure that the data from a single device or cluster-head is not artificially weighted based on the number of cluster-heads it is sent to. For example, if device A has three parents but device B only has two parents and data is sent to and aggregated with a simple averaging function at all parents for each collection period the aggregated data will be artificially weighted based on the number of parents (i.e. after one aggregation; A's data has three copies but B's data only has two). Due to the hierarchical structure this problem could compound as the data is aggregated if the parents have fewer cluster-heads than expected. To

counter the artificial weighting one of the following options can be used: all devices and cluster-heads need to have the same number of parents; a weighted aggregation method such as a weighted average where the weight corresponds to the number of parents; a round-robin data distribution where all of the devices utilize a common sampling rate and send their data to one parent at a time.

1.1.2 Spatiotemporal Multi-Resolution

There are different categories of users when it comes to wireless sensors. Take for example a wireless sensor network that is distributed across a park. The casual user (i.e. a person that came to the park to relax) is interested in finding a comfortable area to sit on the grass in the sun by querying the sensor network through their smart phone. This type of user is typically interested in the closest place that matches the description and as such needs more details on the closer locations and little to no detail on the rest of the park. A park manager on the other hand needs information about the entire park at a broad, low-detailed level unless there is a problem. When a problem does occur, such as a broken sprinkler, the park manager needs to be made aware of the issue and be provided with more detailed information about the issue. Spatial multi-resolution analysis can provide each user with information about the surrounding environment at a specified level of detail Ganesan et al. [2005]

Reliable detection of an abnormal environment relies on a sensor networks ability to determine the difference between the normal operating background and an anomalous event. This can require the sensor network to store information about what is a normal condition and be able to adjust its view of a normal condition based on the natural changes to the environment. For example, a normal operating temperature of a device that is in the sun during the morning might be significantly higher than when the device is in the shade during the afternoon. A statistical analysis of the previous week's data consisting of a temperature for every hour would suffice to determine if this were the case. Likewise, the normal operating temperature of a device in the winter can be anomalous if it is encountered in the summer. The acceptable seasonal range can be determined using less detailed information over a longer range. In this case the previous year or two of data including an average, minimum and maximum for each week is sufficient to determine the seasonal change and current acceptable seasonal range. Temporal multi-resolution analysis can provide the change in detail over time by compressing and aging older data while maintaining more detailed data for recent events Ganesan et al. [2005]

The structure provided by a recursive area hierarchy provides an ideal paradigm for spatiotemporal multi-resolution analysis. Aggregation data at each cluster-head allows an increasing perspective of the network from the lower-tiered cluster-heads that maintain a detailed view of the local environment to the higher tiered cluster-

heads that have a broad view of the network. This aggregation scheme provides a multi-resolution spatial component that can be utilized by applications. Adding an aging mechanism for data stored at each cluster-head where older data is compressed more than recent data provides the temporal multi-resolution component. The spatial compression can be used to address message implosions by reducing the data that is sent to the devices performing higher level data analysis. However at times these devices need more detailed information and must query the lower level devices accordingly. This is where the temporal compression comes into play as it can reduce the amount of data that needs to be stored on any given device while still retaining information that is useful for the higher level devices. The multi-resolution spatiotemporal capabilities of a paradigm such as the recursive area hierarchy can be used to benefit applications such as distributed object tracking Kulathumani et al. [2007], Tsai et al. [2007], Chang et al. [2008] and anomaly detection Xie et al. [2011], as well as many more.

1.1.3 Local Synchronization

In a duty-cycled network, synchronization of neighboring devices is imperative for communication. This is because the clocks and timers that wake up a device are prone to clock drift which can cause two different devices to wake up at different

times. The severity of clock drift depends on factors including temperature, source voltage, and variability in the timing components and circuits. To counter clock drift special messaging that is designed to correlate two or more remote clocks can be used. The frequency of the synchronization messaging depends on the maximum clock drift of the different devices, as well as the maximum allowed time difference.

How often synchronization needs to be performed depends heavily on the end application, the protocols used, and the hardware. A better understanding of the synchronization period can be gained by considering a common clock crystal with an oscillation frequency, $OscFreq$, of 32.768 KHz and a clock accuracy, $OscAcc$, of 20 ppm. The maximum time drift per minute assuming one clock drifts high and the other low can be calculated using formula 1.4.

$$OscDrift = OscFreq/OscAcc \tag{1.1}$$

$$HighOscDrift = OscFreq + OscDrift \tag{1.2}$$

$$LowOscDrift = OscFreq - OscDrift \tag{1.3}$$

$$MaxTimeDriftPerMinute = (HighOscDrift/LowOscDrift * 60) - 60 \tag{1.4}$$

Using these formulas for a 32.768 KHz, 20 ppm crystal the maximum time drift per minute between two separate devices is 2.4 milliseconds. This might not seem

like a significant amount of time but given that the data rate of IEEE 802.15.4, a common mesh networking protocol, is 250 Kbits per second, IEEE [2003] this would be equivalent to sending approximately 75 bytes of data. When dealing with short, infrequent messages the preamble length and/or listening time required to synchronize a message if no other synchronization is performed is sufficient. For example, if a message is sent every 15 minutes the device would need to have a preamble that is at least 36 milliseconds or a listening time (i.e. the amount of time the receiver is listening in a given period) for the receivers that is at least 72 milliseconds for the worst-case scenario. The maximum times stated are only when one solution is used, as the synchronization period and preamble length are inversely related to each other (i.e. if the preamble is longer the listening time can be reduced and vice versa). Further, the worst case is highly unlikely and can introduce twice as much error as is typical Schmid et al. [2010].

If synchronization cannot take place with normal messaging because little to no messaging occurs then minimizing synchronization such that it is performed as infrequently as possible can reduce the synchronization overhead. However, this comes at the cost of potentially increasing the delay and cost to join a network. Take for example a group of devices that synchronize once every 30 minutes where one device is scheduled to send a message 10 minutes after the last synchronization. In order to send the message the sender and receiver need to be able to deal with a possible

24 milliseconds of clock drift at the scheduled transmission. Adding a new device to the same set of devices that are synchronized every 30 minutes would require the new device to listen until the next synchronization period before it can know that there are other devices in the area and can be synchronized with those devices. Some situations can require the network to be synchronized more often in order to reduce latency and reaction time.

1.1.3.1 Local Synchronization Mechanisms

Synchronization can be performed using methods including synchronization beacons/messages, preambles, wake-up signals, low power wake-ups, etc. The "best" solution depends heavily on the situation as each solution has its own benefits and drawbacks. Low power wake-up and wake-up signals allow for devices to be woken up by a remote device without periodic messages. However the transmission of the wake-up in both situations requires a significant amount of power and as such is not always suitable for battery powered devices that must operate for years before the batteries are replaced. Synchronization beacons/messages do not require a power intensive wake-up signal but require periodic communication between a set of devices to ensure accurate timing.

Low power wake-ups range from devices that are completely passive, where the energy required to wake-up the device is supplied by the transmitter, to devices with active low-power components to help increase the range. As such low power wake-ups

consume little to no power at the receiving end but can require a significant amount of power for transmission as the power used to wake-up the device is transmitted over-air. Since the signal strength of a wireless RF transmission does not scale linearly, increasing the range can require a significant increase in the transmission power. The typical range for a low power wake-up is less than ten meters Ansari et al. [2009].

A wake-up signal on the other hand has a larger range, 100+ meters, but requires devices to listen for the wake-up signal at a given period. The length of the wake-up signal must also be longer than the listening period to ensure that all devices will turn on and listen when the wake-up is being transmitted. If for example a wake-up signal is five seconds long then every device must wake-up and listen for the signal at least once every five seconds to ensure it does not miss the wake-up. The different parameters of such a solution can be tuned for different situations. In general the shorter the wake-up is the more often a device must listen for the wake-up which requires more power from the listening device, whereas longer wake-ups require more power for transmission of the wake-up signal Yoon et al. [2008]. Longer wake-ups also generate longer periods of interference and may be subject to transmission regulations enforced by the government (e.g. FCC).

Another way to synchronize devices is to send a message that includes the current time from the transmitter's perspective. Receiving devices can then offset their clocks to be similar to the transmitter's clock. This type of synchronization can

be performed using normal messages (i.e. no specialized signaling). In many cases this can be done with each message that is sent, which helps to reduce the overhead of synchronization. However, if there is a period of limited or no communication then a message might need to be sent for the sole purpose of ensuring synchronization Wang et al. [2012], Buettner et al. [2006], El-Hoiydi and Decotignie [2004].

1.1.3.2 Synchronization Classifications

Synchronization of a set of devices can be classified by which devices are synchronized and also by how the devices are synchronized. Three different classifications for what devices are synchronized are: 1) all neighbors, where a device is synchronized with all of its neighbors; 2) cluster synchronization, where a cluster of devices are synchronized; and 3) pair-wise synchronization, where a device and its communicating pairs are synchronized. Two classifications for how devices are synchronized are synchronous and asynchronous synchronization which dictate when the devices are turned on.

The *all neighbors synchronization* requires a device to be synchronized via one clock with all of its neighbors. This type of synchronization can be required in a duty-cycled network if a device wants to broadcast a message to all of its neighbors with a single transmission. The all neighbors type of synchronization can require that every device transmit synchronization messages. For example, in a sparsely distributed network a device might have six neighbors but only share one or two neighbors with

any other neighbor making it difficult to synchronize all of the nodes in the local area without sending a message.

Cluster synchronization is a step down from the all neighbor solution and only requires a device to be synchronized with its cluster. This type of synchronization can be performed by the cluster-head and does not necessarily require any device other than the cluster-head to transmit a message for synchronization purposes.

Pair-wise synchronization requires that a device and receiver be synchronized for the purposes of communication. In many situations this can be handled by the normal communication between the sender and receiver but could require additional messaging at times of little to no communication.

In a *synchronously synchronized* environment all of the synchronized devices turn on and are ready to communicate at the same time. This type of solution is necessary for a broadcast/flooding situation as all of the neighbors must be listening for the transmission. In a uni-cast environment, where a message is only intended for a single receiver, this type of solution is not ideal. Every device other than the intended receiver uses resources listening to a message that will be discarded. In the case where there is a transmission sent for them at the same time there is the possibility of channel contention that could corrupt one or both messages which would require re-transmission Wang et al. [2012].

Asynchronously synchronized devices are synchronized but do not necessarily

turn on at the same time as the rest of the devices they are synchronized with. This allows for uni-cast messaging to be scheduled at a time when other devices are off and helps to minimize interference and optimize power consumption for uni-cast messaging. However, it can require redundant transmissions to successfully broadcast a message to all neighbors unless there is a way to ensure that all of the neighbors are ready and listening at the same time Wang et al. [2012].

1.2 Related Work

This section provides an overview of two previous approaches for constructing a recursive area hierarchy over a WSN. The first approach, Hierarchical Beaconing (HB), was introduced in 2003 and the other, a gossip-based solution (GB), was introduced in 2009. As defined in the literature, both of these algorithms can construct a one-parent clustering hierarchy on a multi-hop WSN using a bottom-up type solution.

1.2.1 Hierarchical Beaconing

The initial hierarchy construction for Hierarchical Beaconing (HB) consists of recursively building the hierarchy from the bottom-up Du et al. [2008, 2004], PalChaudhuri et al. [2005], Bandyopadhyay and Coyle [2003]. The process starts with selecting $tier^1$ cluster-heads and having those cluster-heads flood a small area of the network,

possibly one to two hops, with a beacon to inform the surrounding devices that there is an available cluster-head. All devices listen to the message and use it to determine which cluster-head is closest to them in terms of the number of hops. $Tier^2$ cluster-heads are then selected and flood an area exponentially larger than the $tier^1$ cluster-heads with their beacon and devices use these beacons to determine which $tier^2$ cluster-heads are the closest. The process continues until the $tier^N$ cluster-head floods the entire network at which point the hierarchy has been constructed and is ready for use. Maintenance follows the same protocol and is run periodically to detect and adapt to changes in the network. While this process in the general case is effective, the requirements necessary to flood a wireless sensor network with duty-cycled devices is an inefficient use of limited resources.

1.2.2 Gossip-Based Solution

Gossip-based protocols on the other hand reduce the overhead of constructing a recursive area hierarchy by removing the flooding algorithms that typically forward a message as soon as it is received Iwanicki and Van Steen [2009], Iwanicki and van Steen [2010]. These protocols instead have devices that periodically broadcast hierarchy information one-hop, listen for all messages from their neighbors, and merge the gathered information with their own before performing another single-hop broadcast.

The hierarchy information is still propagated to all of the devices in the required area, but instead of using flooding it is done using single-hop broadcasts. This significantly reduces the number of messages that are sent but increases the latency of the propagation. For example, in Iwanick's work Iwanicki and van Steen [2010] it is shown that network construction can take more than 5.8 hours when the gossip interval is five minutes. When compared to the lifetime of the network the construction time is insignificant, but is not practical for testing purposes, for applications that need to start gathering data as soon as they are deployed, or for applications that have mobile devices. Further, since the same algorithm is used to maintain the structure, the reaction time to a change in the network is also lagged as the loss of cluster-heads could take almost as long to recover from as it took to construct the network. When dealing with duty-cycled, sparsely distributed networks, there is still the requirement that every device needs to be synchronized with its neighbor, which is an expensive task.

1.3 Naming Conventions

NOA is the name of the proposed algorithm used to create a multi-parent recursive area hierarchy as defined in §3. There are a number of different variants that were utilized when evaluating NOA including different combinations of: two-parent and

three-parent variants indicated by the superscripts NOA^2 and NOA^3, respectively; 1-Cluster-Head and N-Cluster-Head variants indicated by the subscripts NOA_{1-CH} and NOA_{N-CH}, respectively. For example, the variant including two parents and $1-CH$ is noted as NOA^2_{1-CH}. The notation for hierarchical beaconing solutions follow a similar style and are indicated by HB followed by super-scripting for the number of parents: HB^1, HB^2 and HB^3. Sub-scripting is used to indicate $1-CH$ or $N-CH$: HB_{1-CH} and HB_{N-CH}.

1.4 Overview

Chapter 2 will provide details about the basic network model as well as introduce some formulas for calculating the cost of performing different tasks given the basic model. Details about the proposed protocol, NOA, are presented in chapter 3. The simulation environment including the design and how it was used is covered in chapter 4. An evaluation and comparison of the network model for NOA, hierarchical beaconing, and gossip-based, protocols as well as evaluation and comparison of the network model and the simulation environment for NOA and hierarchical beaconing is provided in chapter 5. Concluding remarks can be found in chapter 6.

CHAPTER 2. NETWORK MODEL

The network consists of a set of devices that are sparsely populated (approximately 6 neighbors per device) across a vast geographic region thus requiring multi-hop communication between devices. There is no global synchronization between devices, and as such devices that communicate with each other need to be locally synchronized to perform duty-cycling of a device's radio.

This chapter provides an overview of the ideal network organization and distribution as well as some basic cost models associated with hierarchy construction, data aggregation and synchronization. An overview of the basic configuration and the ideal distribution is provided in §2.1. Examples of ideal one-, two- and three-parent hierarchies are presented in §2.2. Details about the hierarchy construction model are provided in §2.3. The aggregation cost model is described in §2.4. Details about synchronization requirements and the associated communication cost are presented in §2.5.

2.1 Configuration Algorithms and Ideal Distribution

Autonomous configuration algorithms construct the hierarchy in a recursive manner starting from the individual device, also referred to as $tier^0$ clusters. $Tier^1$

clusters consist of devices within one hop of the $tier^1$ cluster-head. At some level there exists a $tier^N$ with three or fewer devices acting as the highest level cluster-heads that encompass the entire network. $Tier^{n+1}$ clusters, where $0 \leq n \leq N$, are super-clusters composed of adjacent $tier^n$ clusters. A $tier^n$ cluster can be part of three or fewer super-clusters depending on the number of parents allowed in the specified hierarchical structure (one-, two-, or three-parents). The 1-Cluster-Head, 1-CH, model allows a cluster-head to act as a single cluster-head at only one tier, whereas the N-Cluster-Head, N-CH, model allows a cluster-head to act as one cluster-head per tier. The radius of a cluster is the number of hops from the center of the cluster to the cluster's edge.

The ideal distribution of the network is one where every device is equally spaced and has six neighbors excluding edge devices. This device distribution can be achieved by placing a hexagonal overlay over the entire region and placing a device in the center of each hexagon. The radius of the hexagon, r, depends on the reliable communication range of each device, $comm\_range$, with $\sqrt{3}r <= comm\_range < 3r$ (Fig. 2.1).

2.2 One-, Two, and Three-Parent Hierarchies

The hierarchies presented are constructed on an ideally distributed network as previously described. The example hierarchies and formulas presented in this

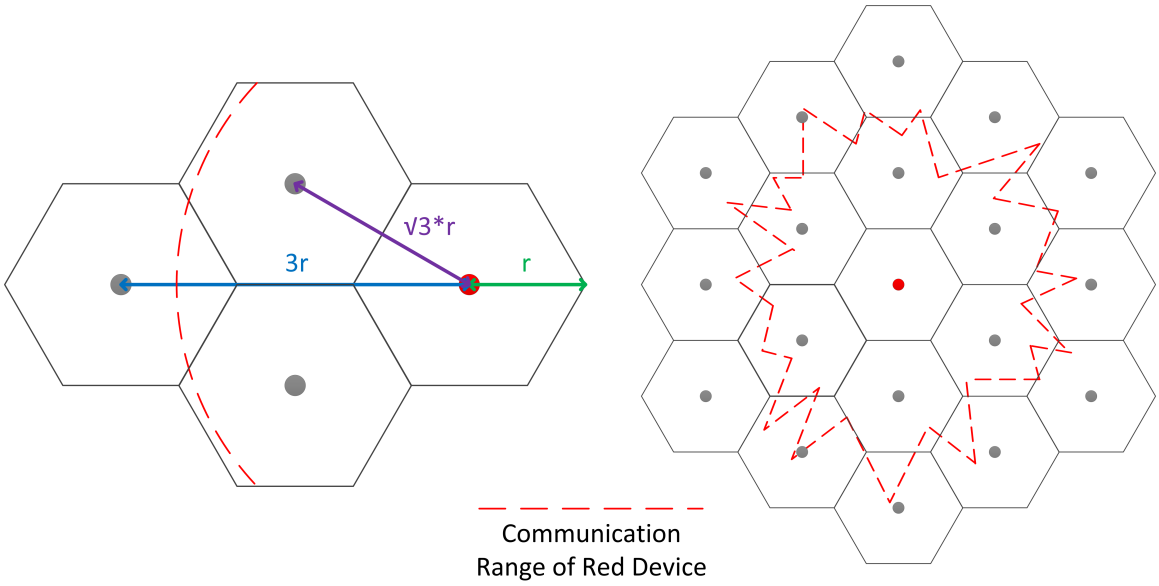
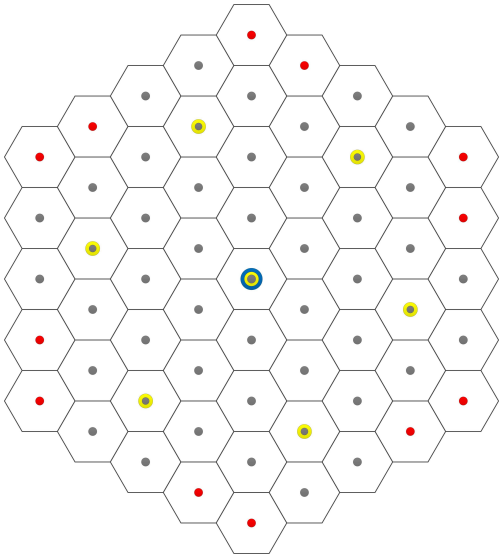
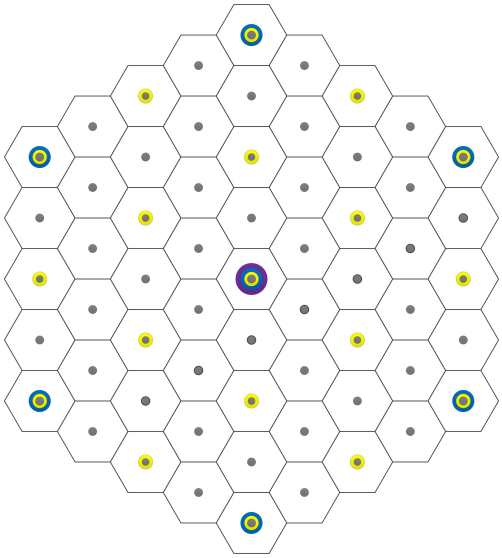


Figure 2.1: The ideal device distribution where each device can communicate directly with its six neighbors. The red dashed line is an illustration of the required signal strength to ensure communication with the neighboring hexagonal cells.

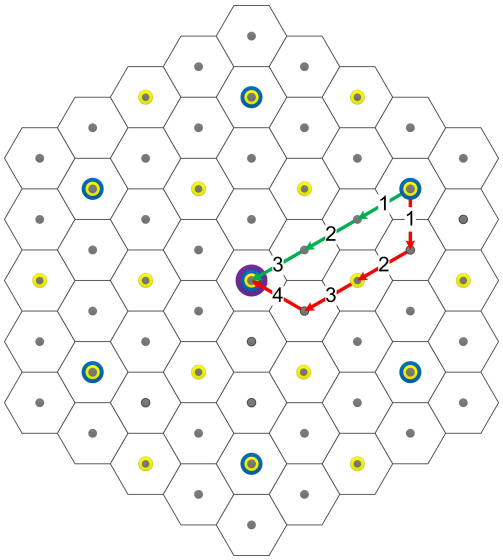
section assume an N-CH type hierarchy with cluster-heads centered in their respective clusters. In the case of $1-CH$ type hierarchies the formulas need to take into account the cost of electing a descendant as the next tier's cluster-head and the fact that a cluster-head could exist on the edge of a cluster as opposed to the center of the cluster as assumed in the N-CH model.



(a) One-Parent



(b) Two-Parent



(c) Three-Parent

Figure 2.2: Different hierarchies configured using the ideal layout of devices. The smallest gray circles represent $tier^0$ devices, yellow circles are $tier^1$ cluster-heads, blue circles are $tier^2$ cluster-heads, and purple circles represent $tier^3$ cluster-heads. The red circles represent devices that are not part of a cluster in the ideal hierarchy and require special handling. In c) there are two routing paths from a child to a parent shown; one in red, the other in green.

2.2.1 One-Parent Hierarchy (Non-NOA)

The One-parent hierarchy (Fig. 2.2(a)) requires that clusters are part of only a single super-cluster. As such, this hierarchy cannot be constructed using the proposed protocol, NOA. It is displayed for comparison because it is what is primarily constructed using the state-of-the-art methods. Formulas 2.1-2.5 can be used to: determine the number of tiers in a network with n nodes, the number of nodes in a tier t cluster, the radius of a cluster at $tier^t$, the distance in hops between a $tier^t$ parent and its $tier^{t-1}$ child, and the percent of nodes that act as a $tier^t$ cluster-head where $0 <= t <= numberOfTiers$. These formulas assume the ideal distribution as shown in Fig. 2.2 and an N-CH hierarchy with cluster-heads centered in their respective clusters.

Let, n = number of nodes and t = tier of the cluster or parent

$$\#OfTiers = \text{ceiling}(\log_7(n)) \tag{2.1}$$

$$\#OfNodesInClusterTier = 7^t \tag{2.2}$$

$$RadiusOfCluster = (3^t - 1)/2 \tag{2.3}$$

$$numOfHopsFromParentToChild = 3^{t-1} \tag{2.4}$$

$$percentOfNodesAsCHs = 7^{-t} * 100 \tag{2.5}$$

2.2.2 Two-Parent Hierarchy

The two-parent hierarchy (Fig. 2.2(b)) allows clusters to be part of at most two super-clusters. In the ideal case, every device that is not on the network edge has six neighbors and two-parents. Siblings cluster-heads in this situation have one child in common. Since there can be multiple top level devices, $tier^N$ devices report to each other. Formulas 2.6-2.10 can be used to: determine the number of tiers in a network with n nodes, the number of nodes in a tier t cluster, the radius of a cluster at $tier^t$, the distance in hops between a $tier^t$ parent and its $tier^{t-1}$ child, and the percent of nodes that act as a $tier^t$ cluster-head where $0 \leq t \leq numberOfTiers$. These formulas assume an ideal distribution as shown in Fig. 2.2 and an N-CH hierarchy with cluster-heads centered in their respective clusters.

Let, n = number of nodes and t = tier of the cluster or parent

$$\#OfTiers = \text{ceiling}(\log_2(1/6 * (\sqrt{3} * \sqrt{4 * n - 1} + 3))) \tag{2.6}$$

$$\#OfNodesInClusterTier = 3 * (2^t)^2 - 3 * 2^t + 1 \tag{2.7}$$

$$RadiusOfCluster = 2^t - 1 \tag{2.8}$$

$$numOfHopsFromParentToChild = 2^{t-1} \tag{2.9}$$

$$percentOfNodesAsCHs = 4^{-t} * 100 \tag{2.10}$$

2.2.3 Three-Parent Hierarchy

Clusters in three-parent hierarchies (Fig. 2.2(c)) can be part of at most three super-clusters. As with the two-parent hierarchy if there are multiple $tier^N$ cluster-heads these cluster-heads report to each other. In the ideal case every non-edge cluster-head six children and three parent. Sibling cluster-heads have two children in common. In the three-parent variants the number of hops between a parent and a child depends on the routing solution.

Formulas 2.11-2.14 can be used to: determine the number of tiers in a network with n nodes, the number of nodes in a tier t cluster, the radius of a cluster at $tier^t$, and the percent of nodes that act as a $tier^t$ cluster-head where $0 \leq t \leq numberOfTiers$. The distance in hops between a $tier^t$ parent and its $tier^{t-1}$ child can be calculated using formulas 2.15 and 2.16 depending on which routing scheme is used. All formulas assume an ideal distribution as shown in Fig. 2.2 and an N-CH hierarchy with cluster-heads centered in their respective clusters.

Let, n = number of nodes and t = tier of the cluster or parent

$$numberOfTiers = \begin{cases} 1, & 0 < n \leq 7 \\ 2, & 7 < n < 12 \\ \text{ceiling}(\log_2(2/9 * (\sqrt{3} * \sqrt{4 * n + 3} - 3))), & n \geq 12 \end{cases} \tag{2.11}$$

$$\#OfNodesInClusterTier = \begin{cases} 1, & t = 1 \\ 9 * (3 * (2^{t-2})^2 + 2^{t-2}) - 6t + 7, & t > 1 \end{cases} \tag{2.12}$$

$$RadiusOfCluster = \begin{cases} 1, & t = 1 \\ 3 * 2^{t-2}, & n > 1 \end{cases} \tag{2.13}$$

$$percentOfNodesAsCHs = 3^{-t} * 100 \tag{2.14}$$

A message that is routed from a cluster-head to its parent by traversing the hierarchy is shown in Fig. 2.2(c) as the red path. In this case all communication from $tier^n$ to $tier^{n+1}$ passes through the common $tier^{n-1}$ child. For example, if a $tier^2$ device is trying to communicate with a $tier^3$ device the communication would need to pass through a common $tier^1$ child. Likewise communication between $tier^2$ devices and $tier^1$ children would pass through a common $tier^0$ child. Formula 2.15

can be used to determine the number of hops required to send a packet from a $tier^t$ parent to a $tier^{t-1}$ child given an ideal distribution.

$$numOfHopsFromParentToChild = 2^{t-1} \tag{2.15}$$

If the routing table allows a message to take a more direct route to the destination, a lower cost solution in terms of the number of hops can be provided. The lower cost route is shown in Fig. 2.2(c) as the green path. For example if a $tier^n$ device is trying to communicate with a $tier^{n+1}$ device there is no need to communicate through the $tier^{n-1}$ device as in the previous example. Formula 2.16 can be used to determine the number of hops required to send a packet from a $tier^t$ parent to its $tier^{t-1}$ child given an ideal distribution.

$$numOfHopsFromParentToChild = \begin{cases} 3^{(t-1)/2}, & \text{if } t \text{ is odd} \\ 2 * 3^{(t-2)/2}, & \text{if } t \text{ is even} \end{cases} \tag{2.16}$$

2.2.4 Comparison of the One-, Two-, and Three-Parent Hierarchies

Figures 2.3, 2.4, 2.5, and 2.6 visually compare the difference between a one-, two- and three-parent hierarchy with regard to various properties including: the num-

ber of tiers, radius of a cluster, percent of nodes that act as cluster-heads and the total number of cluster-heads in an N-tiered network. Looking at each figure by itself shows that there are significant differences between the one-, two- and three-parent hierarchies with respect to these properties. Figure 2.3 shows that one-parent hierarchies can support more devices with fewer tiers. Two- and three-parent hierarchies have a smaller cluster radius (Fig. 2.4). This difference is such that even higher tiered two- and three-parent cluster radii are shorter than lower tiered one-parent cluster radii. One-parent hierarchies require a smaller percentage of devices that act as cluster-heads at a given tier than in the two- parent case and the same can be said of the two- and three-parent cases (Fig. 2.5). Figure 2.6 shows that a one-parent hierarchy of an equivalent tier requires more cluster-heads than the two- and three-parent hierarchies. However, the number of nodes in a $tier^N$ network must also be considered as the two- and three-parent hierarchy of the same tier covers fewer devices.

This is only one case where looking at one figure by itself can incorrectly provide favor towards the number of parents in a hierarchy compared to others. Take for example Figure 2.3. The one-parent hierarchy requires only six tiers for one hundred thousand devices whereas the two- and three-parent hierarchies require eight tiers and as such when viewing this information by itself it seems that the one-parent hierarchy is better. However, all four of the figures previously mentioned are intertwined.

Consider the number of tiers required for a network (Fig. 2.3) and the radius of a cluster (Fig. 2.4). A network of one hundred thousand devices requires six tiers for the one-parent hierarchy and eight tiers for the two- and three-parent hierarchies. However, the radius of a one-parent, $tier^6$ cluster is around 400 hops, whereas the radius of the two- or three-parent $tier^6$ cluster is approximately 250 hops or 195 hops, respectively. The algorithms that are used to manage the network can affect each of the various correlations between these properties in different ways, which increases the complexity of directly comparing these charts.

2.3 Hierarchy Construction Cost Model

The cost to bootstrap a hierarchy in terms of the number of message hops that are required is presented for three different techniques: hierarchical beaconing, gossip based, and the proposed protocol NOA. Each solution has its own model due to the different techniques that are utilized during hierarchy configuration.

2.3.1 Hierarchical Beaconing

Hierarchical beaconing uses flooding to propagate the hierarchy topology. In order to configure a cluster-head, a message containing information about the new cluster-head is flooded from the cluster-head to the edge of the cluster. As such

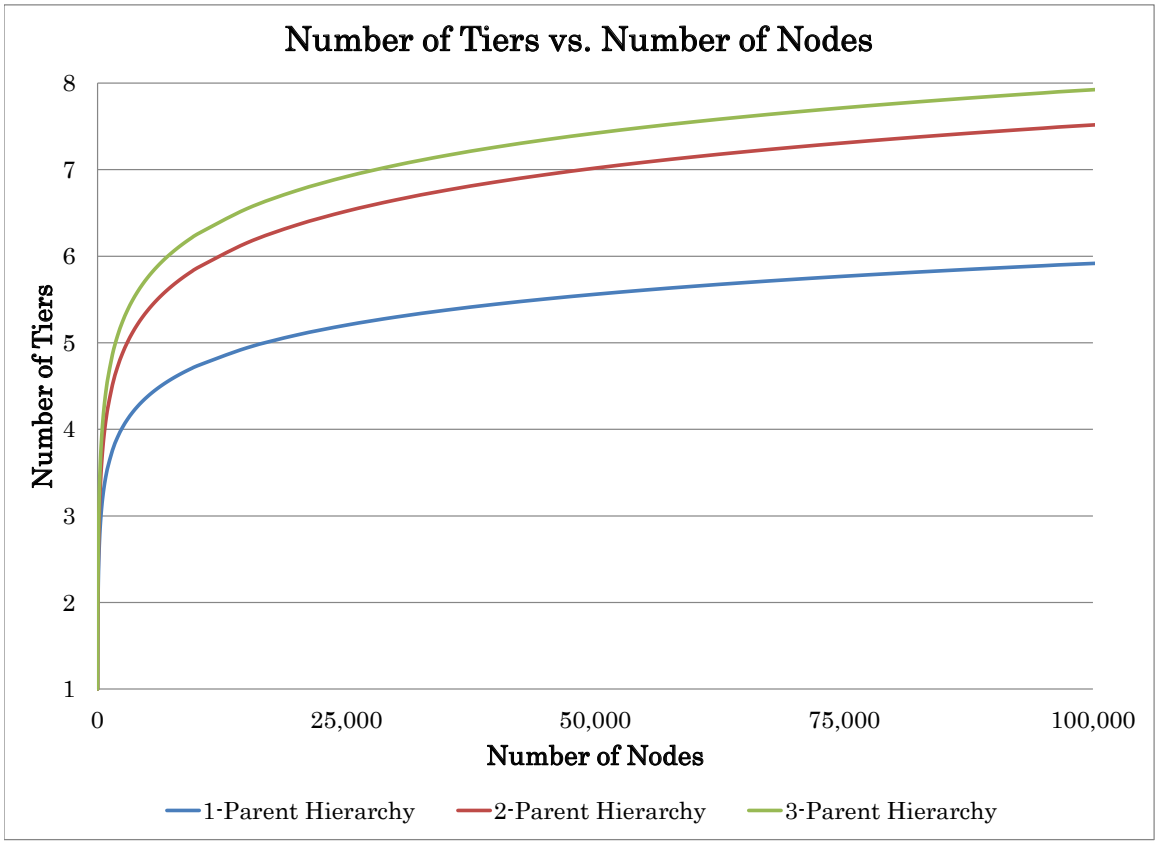


Figure 2.3: Comparison of the number of nodes in a hierarchy with the number of tiers in a one-, two-, and three-parent hierarchy assuming the ideal network distribution.

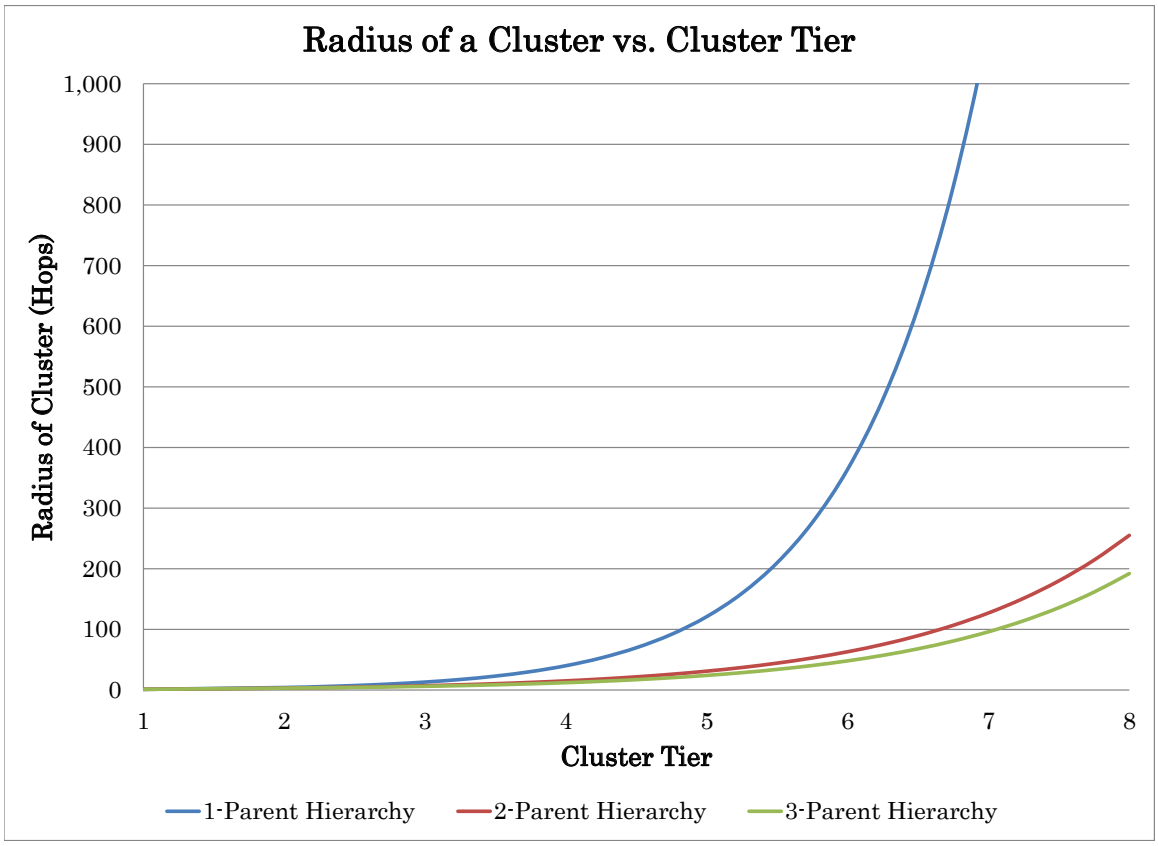


Figure 2.4: Comparison of the tier of a complete cluster in a one-, two-, and three-parent hierarchy with the radius of said cluster. This comparison assumes that a cluster at tier-n is complete and has the number of nodes as shown in Figure 2.3

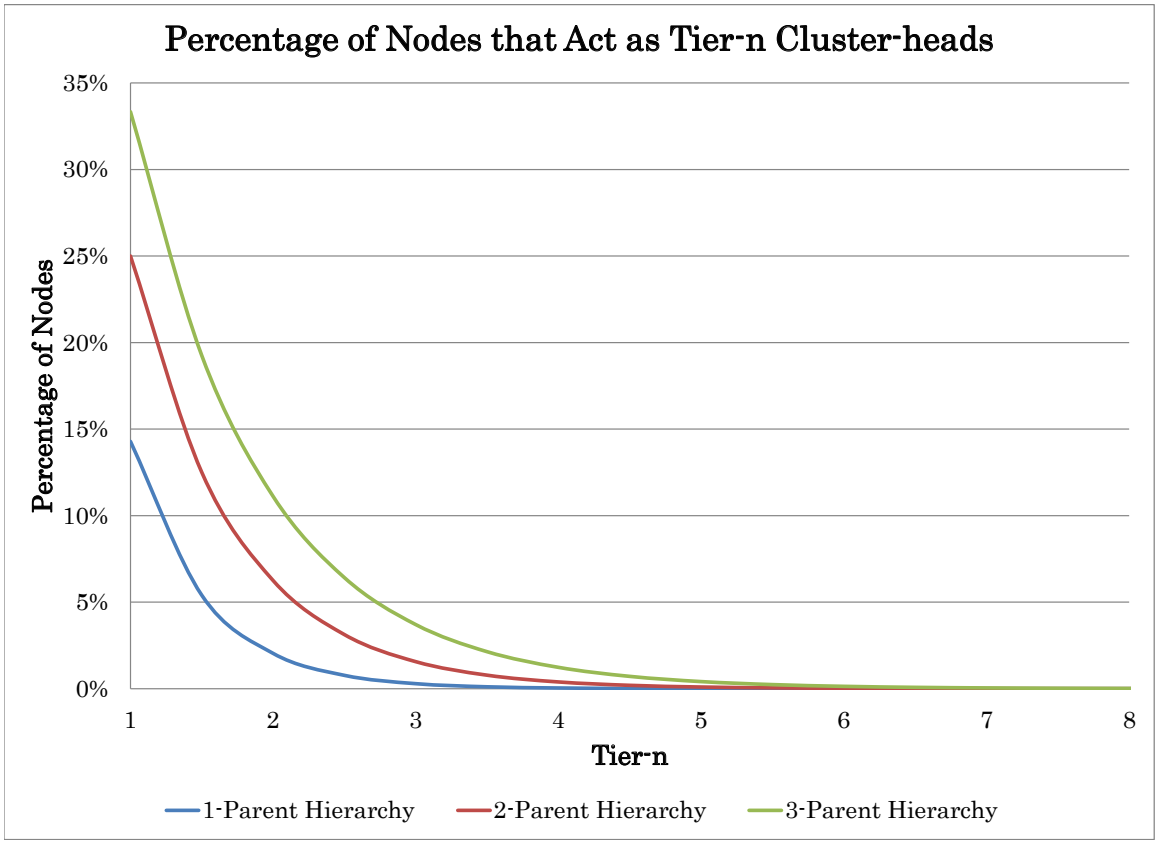


Figure 2.5: The percentage of nodes that act as a cluster-heads at a given tier.

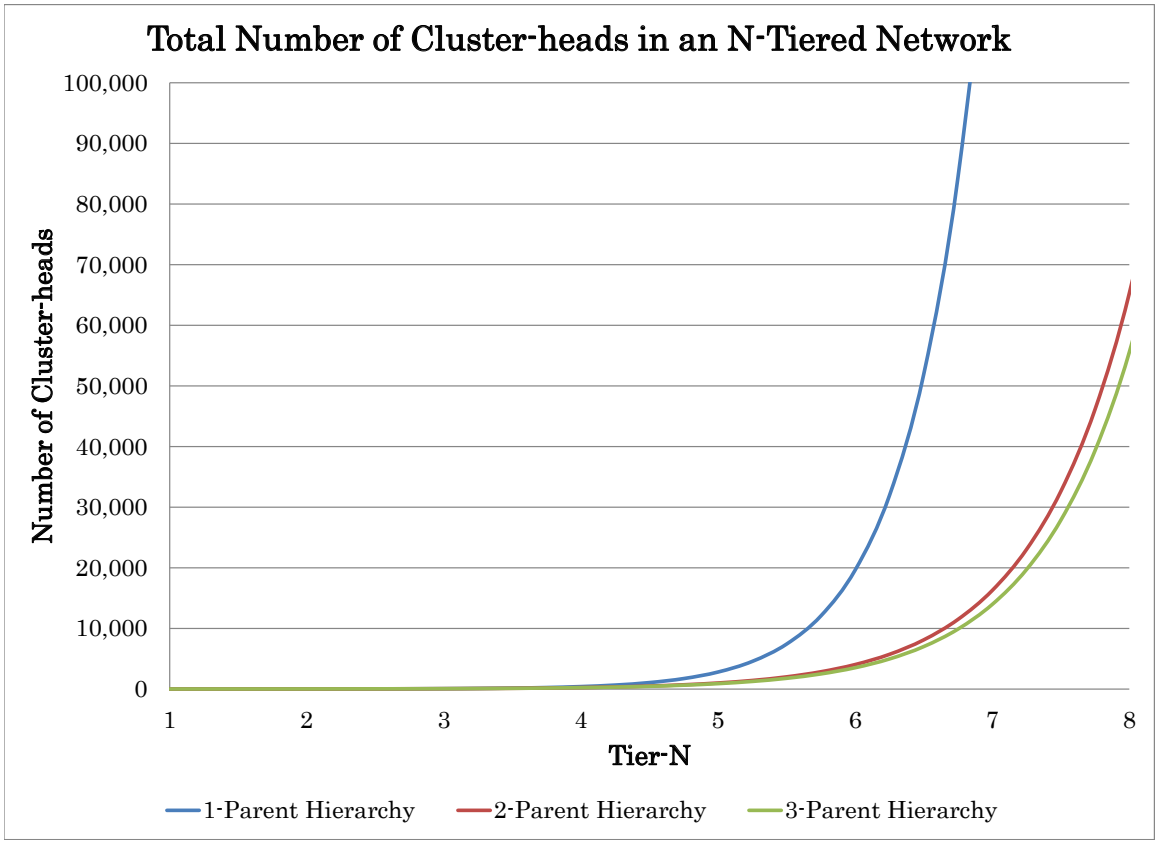


Figure 2.6: The total number of cluster-heads in an N-tiered network. A device that is acting as a cluster-head at multiple tiers is counted multiple times, once for each tier.

$RadiusOfCluster(t)$ can be use to determine the number of hops a message needs to be flooded. $RadiusOfCluster(t)$ can be calculated using formula 2.3, 2.8 or 2.13 depending on the number of parents the hierarchy has.

There are different flooding mechanisms that can be used, two of which are considered below. The first flooding solution would be to assume that every message that is received is re-broadcast until the hop counter equals 0. In this situation a message can be re-broadcast by the same device multiple times which significantly increases the cost. The cost to configure a cluster-head of tier t using this type of flooding can be calculated using formula 2.17. The total cost for configuring the network using this approach can be calculated using formula 2.19, where T represents the highest tier in the network and can be calculated using formula 2.1, 2.6 or 2.11 depending on the number of parents. $PercentOfNodesAsCH(t)$ which is used in the previous formulas can be calculated using formula 2.5, 2.10 or 2.14 depending on the number of parents.

$$HopsToConfigCH(t) = (6^{RadiusOfCluster(t)} - 1)/5 \tag{2.17}$$

$$CHsPerTier(t) = \#OfNodes * PercentOfNodesAsCH(t) \tag{2.18}$$

$$\#OfHops = \sum_{t=1}^{t \leq T} (CHsPerTier(t) * HopsToConfigCH(t)) \tag{2.19}$$

The second flooding protocol limits a device such that it can only broadcast each message once. This can be done by including a sequence number in the message and having each device store the sequence number and original sender for any message it forwards. This way if the device receives a message from the same original sender with a sequence number less than or equal to the one that was stored the device can assume that it already sent the message and does not need to re-broadcast. This basic solution can have issues with messages being received out of order but that is outside of the scope of this work. In this situation the configuration cost for a cluster-head of tier t can be calculated using formula 2.20. The total cost for configuring the network can be calculated using formula 2.22, where T represents the highest tier in the network. $PercentOfNodesAsCH(t)$ can be calculated using formula 2.5, 2.10 or 2.14, depending on the number of parents.

$$HopsToConfigCH(t) = 3 * (RadiusOfCluster(t)^2 - RadiusOfCluster(t)) + 1 \tag{2.20}$$

$$CHsPerTier(t) = \#OfNodes * PercentOfNodesAsCH(t) \tag{2.21}$$

$$\#OfHops = \sum_{t=1}^{t \leq T} (CHsPerTier(t) * HopsToConfigCH(t)) \tag{2.22}$$

2.3.2 Gossip-Based

The gossip-based solution Iwanicki and van Steen [2010] propagates hierarchical information using periodic single hop broadcasts that merge the information from neighboring devices before re-broadcasting it. As such the cost to configure a single cluster-head is similar to flooding except that instead of re-broadcasting every message as soon as it is received, multiple messages are merged prior to re-broadcasting.

Every device must broadcast a message once per period until the highest tiered cluster-head is configured and propagates its information to all of its neighbors. As such the number of one-hop messages sent for construction can be calculated by multiplying the number of rounds it takes to construct the hierarchy by the number of nodes in the network (formula 2.24). The number of rounds required for hierarchy construction can be calculated by taking the sum of how many rounds it takes for each $tier^t$ cluster to configure. This is because $tier^{t-1}$ cluster-heads must be configured before $tier^t$ cluster-heads start. In order to configure the $tier^t$ cluster-heads a message must propagate to the edge of the cluster and as such it takes $RadiusOfCluster(t)$ rounds per cluster. Thus, the number of construction rounds required is equal to the sum of the radius of each tier from $1 <= t <= T$, (formula 2.23), where T represents the highest tiered cluster-head.

$$\#ConstuctionRounds = \sum_{t=0}^{t<=T} (RadiusOfCluster(t)) \tag{2.23}$$

$$\#OfHops(t) = \#OfNodes * \#ConstuctionRounds \tag{2.24}$$

2.3.3 NOA - Proposed Protocol

The proposed protocol distributes hierarchy information to neighboring cluster-heads through common children using uni-cast/multi-cast messaging. For simplicity this model assumes uni-cast messaging. In order to configure a cluster-head a message must be sent from the new cluster-head to all of its children. When a child receives a message from a new parent it waits before responding. In the best case the child will receive a message from all of the other potential parents in the area before responding to the first parent. This makes it to where the child has to only respond once to each parent. In the worst case the device must send a message to every parent each time it receives a new message: one message for the first parent, two for the second (one to update the first parent and one to respond to the second parent), and three for the third parent. The cost of configuring a cluster-head given the best case scenario can be calculated using formula 2.26 and the worst case using formula 2.29. The total cost for configuring the network can be calculated using formula 2.30, where T represents

the highest tier in the network. $PercentOfNodesAsCH(t)$ can be calculated using formula 2.5, 2.10 or 2.14, depending on the number of parents.

$$\#OfMsgs_{BC} = 6 * 2 \tag{2.25}$$

$$HopsToConfigCH_{BC}(t) = \#OfMsgs_{BC} * numOfHopsFromParentToChild(t) \tag{2.26}$$

$$Ave\#OfReplies_{WC} = (\#OfParents + 1) * (\#OfParents + 2) \tag{2.27}$$

$$\#OfMsgs_{WC} = 6 + Ave\#OfReplies_{WC} \tag{2.28}$$

$$HopsToConfigCH_{WC}(t) = \#OfMsgs_{WC} * \#OfHopsFromParentToChild(t) \tag{2.29}$$

$$CHsPerTier(t) = \#OfNodes * PercentOfNodesAsCH(t) \tag{2.30}$$

$$\#OfHops = \sum_{t=1}^{t \leq T} (CHsPerTier(t) * HopsToConfigCH(t)) \tag{2.31}$$

2.4 Aggregation Cost Model

The cost to aggregate a single byte of data depends on whether a device sends the data to all of its parents every round or uses a round robin approach. If a device sends its data to every parent, every round the aggregation cost at a $tier^t$ device is the

number of parents it has, $\#Parents$, times the number of hops from the child to its parent, $\#OfHopsFromParentToChild(t+1)$ (formula 2.33). The aggregation cost in hops for the entire network can be calculated by summing the cost for each device and cluster-head in the network. This is shown in formula 2.34, where T represents the highest tier in the network. $PercentOfNodesAsCH(t)$ can be calculated using formula 2.5, 2.10 or 2.14, depending on the number of parents.

$$CHsPerTier(t) = \#OfNodes * PercentOfNodesAsCH(t) \tag{2.32}$$

$$HopsPerDeviceOrCH(t) = \#Parents * \#OfHopsFromParentToChild(t+1) \tag{2.33}$$

$$\#OfHopsSingleAggregation = \sum_{t=0}^{t \leq T} (CHsPerTier(t) * HopsPerCH(t)) \tag{2.34}$$

Utilizing the round robin approach a device only sends data to one parent every round changing parents with each round. As such the cost to aggregate a $tier^t$ device is $\#OfHopsFromParentToChild(t+1)$ each round. The hierarchy aggregation cost can be determined by summing the cost of every device and cluster-head in the network (formula 2.37). $PercentOfNodesAsCH(t)$ can be calculated using formula 2.5, 2.10 or 2.14, depending on the number of parents.

$$CHsPerTier(t) = \#OfNodes * PercentOfNodesAsCH(t) \tag{2.35}$$

$$HopsPerDeviceOrCH(t) = \#OfHopsFromParentToChild(t+1) \tag{2.36}$$

$$\#OfHopsSingleAggregation = \sum_{t=0}^{t \leq T} (CHsPerTier(t) * HopsPerCH(t)) \tag{2.37}$$

2.5 Synchronization

Synchronization of duty-cycled devices is imperative for communication. This is to make sure that all of the devices that are participating in a communication (i.e. the transmitter and any receivers) are ready to communicate at the same time. Synchronization requirements are driven by different factors including but not limited to: system specifications and requirements, requirements from higher level protocols, device mobility, and network distribution.

Flooding as used in hierarchical beaconing Du et al. [2008], or propagating data via one hop broadcasts as utilized by the gossip based protocols Iwanicki and van Steen [2010] can require a device and all of its neighbors to be ready to communicate at the same time. In a sparse network this can require every device to synchronize with all of its neighbors such that they all turn on at the same time. In the case where a device has six neighbors this can require a device to broadcast one and receive six

synchronization messages per round.

Uni-cast and multi-cast messaging only requires the sender and group of receivers to be on at the same time. This provides additional flexibility and reduces the synchronization requirement such that a transmitting device only needs to be synchronized with the nodes it needs to communicate with.

Synchronization in a multi-parent clustering hierarchy constructed using uni-cast messaging can be performed by having the $tier^1$ cluster-heads broadcast a synchronization message to their children. As the broadcast is only performed by the $tier^1$ cluster-heads and not re-broadcast by the children, a cluster type synchronization is all that is needed to construct and maintain the hierarchy as well as perform local synchronization in the network. In this example, cluster-heads would have to send a message every period and a non-cluster-head device would need to listen to two or three synchronization messages depending on the number of parents it has, which reduces the cost of performing synchronization from seven messages per round to three or four. The cluster-based synchronization example is provided for simplicity and the multi-parent hierarchy could maintain local synchronization using pair-wise synchronization mechanisms but the overhead can vary depending on the situation.

CHAPTER 3. NOA: MULTI-PARENT RECURSIVE AREA HIERARCHY MANAGEMENT PROTOCOL

NOA is an autonomous management protocol for sparsely-distributed, large-scale, wireless sensor networks. The management protocol autonomously configures and maintains a multi-parent structure that follows the recursive area hierarchy paradigm. Cluster-heads utilize common children to determine the location and routing paths to neighboring cluster-heads (i.e. adjacent cluster-heads of the same tier). This step is crucial to the construction process as it allows cluster-heads to know where siblings are located. As such when a sibling elects a parent, said parent can be informed of routes to potential children. This solution is unique compared to the current state-of-the-art protocols in which a parent needs to find potential children by flooding a message to every device in its cluster. Using common children, NOA limits the number of devices that must receive the cluster-head information. This reduces the amount of stateful data stored on each device and reduces the number of messages required for hierarchy construction and network maintenance. Further, NOA removes the need for flooding or propagation of data via one-hop broadcasts, thus requiring a less intensive form of synchronization such as cluster or pair-wise synchronization.

3.1 NOA: Multi-Parent Hierarchy Construction

Bootstrapping the hierarchy starts with selecting $tier^1$ cluster-heads. If the network contains devices that are aware of their relative location in the network then the cluster-head selection algorithm (§3.2) is used. Otherwise, $tier^0$ devices that do not have a cluster-head randomly choose to become a $tier^1$ cluster-head. $Tier^1$ cluster-heads send a message to their (one-hop) neighbors to inform them of a possible parent. $Tier^0$ neighbors upon receipt of the message determine whether they need another parent and after a specified period send a reply acknowledging that they will be a child of the $tier^1$ cluster-head. The reply is also used to inform the parent of any other parents that the child is connected to (Fig. 3.1). If a device accepts an additional parent after the reply to a previous parent was sent, then the device sends another message to inform any previous parents of the newest parent(s). At this point $tier^1$ cluster-heads know of their siblings (i..e. other $tier^1$ cluster-heads that are at most two hops away). As such, when a $tier^1$ cluster-head decides to become or nominate a $tier^2$ cluster-head, it already has a list of potential children and how to contact them. Additional cluster-heads that were created through the random process but are not necessary can be culled and any children from the culled cluster-heads can be passed to surviving cluster-heads.

After the $tier^1$ cluster-heads are configured, as long as a cluster-head has at least

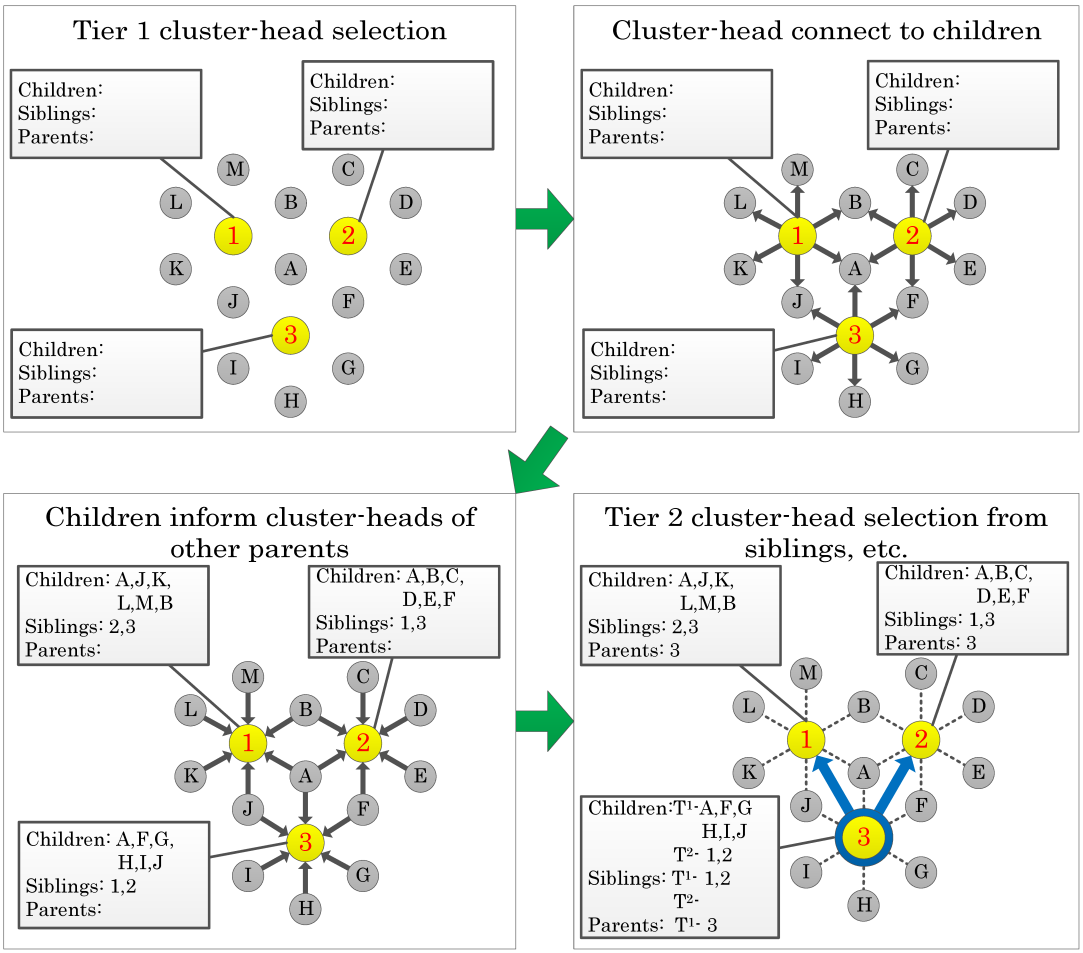


Figure 3.1: An example of how NOA constructs a multi-parent recursive area hierarchy

four children and less than the maximum number of parents, the cluster-head selection process continues. Once a cluster-head is selected the device contacts its potential children and binds with them. After a specified time the children respond, accepting the new parent while also sending along information about any other parents they are linked to. The device also sends a message about a new parent to any other parent it has already responded to. Excessive cluster-heads are culled and children are adjusted as necessary. The process starts over again where viable $tier^n$ cluster-heads randomly choose to become or nominate a $tier^{n+1}$ cluster-head. Only $tier^n$ cluster-heads with fewer than the maximum number of parents and at least four children will become or nominate a $tier^{n+1}$ cluster-head. This limits the cluster-head selection process so that it does not continue infinitely. However, in the multi-parent hierarchies there should be multiple cluster-heads at the highest tier to provide redundancies at the highest level, where each sibling monitors the others as if they were children. This allows the cluster-heads to share information about each other and also allows them to detect if there is a cluster-head loss at the top-most level.

3.2 Cluster-head Selection with Location Aware Devices

When devices know their relative location, in reference to a common point, the cluster-heads can be selected using the algorithms described below. These algorithms,

displayed in a program-like form in §3.2.1-3.2.3, are used to determine which devices act as cluster-heads at each tier in the N-CH solution. The $1-CH$ solution uses these algorithms to determine which devices will become $tier^1$ cluster-heads and which of those $tier^1$ cluster-heads will elect a $tier^2$ cluster-head. The $tier^1$ devices that are used to elect cluster-heads forward a maximum tier value during the election of the $tier^2$ cluster-head. This allows the $tier^2$ cluster-heads to determine if they need to elect a $tier^3$ cluster-head. The process continues until the $tier^N$ cluster-heads are selected. While this algorithm is used to start the cluster-head selection process it does not restrict any device from becoming a cluster-head as needed (i.e. a device has at least four children and not enough parents).

The initial x and y values can be offset to allow for a simple way to shift the location of each cluster-head but still keep the overall structure. Shifting the cluster-heads can result in a different number of cluster-heads at each tier as there are special edge cases that need to be considered. For example, if before a shift there are five $tier^N$ cluster-heads then after a shift there might be three $tier^N$ cluster-heads and a $tier^{N+1}$ cluster-head. This all has to do with where the highest cluster-head is located on the grid. In the case where $x = 0$ and $y = 0$ the highest tiered cluster-head will be in the corner positioned at (0,0) for the single-parent case and (0,1) for the two- and three-parent cases.

3.2.1 Single-Parent Cluster-head Selection

```
x = horizontal cell index of cluster center
y = vertical cell index of cluster center
tier = tier level of cluster-head to be elected
x=x1*2+y%2;
if(tier==0)
  return true;
if(tier%2==1)
{
  Pow7 = pow(7,(tier+1)/2));
  if(y%(Pow7/7)==0  &&
     x%(2*Pow7)==(y*3)%(2*Pow7))
    return true;
}
else
{
  Pow7 = pow(7,floor((tier)/2));
  if((x%(2*Pow7)==0 && y%(2*Pow7)==0) ||
     (x%(2*Pow7)==Pow7 && y%(2*Pow7)==Pow7) )
    return true;
}
return false;
```

3.2.2 Two-Parent Cluster-head Selection

```
x = horizontal index of cluster center
y = vertical index of cluster center
tier = tier level of cluster-head to be elected
mody = pow(2, tier+1);
modx = pow(2, tier);
if((y%mody==1 && x%modx==0) ||
    (y%mody==mody/2+1 && x%modx==modx/2))
  return true;
return false;
```

3.2.3 Three-Parent Cluster-head Selection

```
x = horizontal index of cluster center
y = vertical index of cluster center
tier = tier level of cluster-head to be elected
tiery=(int)(tier/2)+((tier%2)==0)?1:0;
tierx=(int)(tier/2)+((tier%2)==0)?0:1;
mody = 2*pow(3, tiery);
modx = pow(3, (tierx-2)<0?0:(tierx-2));
if((y%mody==1 && x%modx==0) ||
    (y%mody==mody/2+1 &&
    (modx==1 || x%modx==(modx+1)/2)))
  return true;
return false;
```

3.3 NOA: Multi-Parent Hierarchy Maintenance

Cluster-head replacement for a device that recognizes it is almost out of power involves first finding a replacement for the cluster-head and then informing its children, parents and siblings of the replacement. The replacement cluster-head then connects to the children and parents accordingly. Detecting the sudden loss of a device due to any number of reasons (e.g. damage, interference/shielding, or moving a device) requires a periodic communication between child and parent. A simple heart-beat, sent periodically, can suffice. Ideally the heart-beat also contains useful data that needs to be forwarded up the hierarchy, but this is not necessary; either way the solution is the same. Detection of a lost device is performed by the parent.

A $tier^{n+1}$ device determines that a $tier^{n}$ child is lost if it has not received a message from the child after a given period of time. The timeout period needs to be sufficiently longer than the heartbeat period. For example, if the heartbeat period is p seconds then the parent waits $3 * p$ seconds, which should be enough time for three messages to be received. The parent(s) that detects the lost child randomly chooses to select a replacement, thus limiting the number of parents involved with finding a replacement. Once a parent decides to select a replacement it does so and sends out a replacement notification that includes information about both the lost cluster-head and the replacement cluster-head to its siblings and children. Any device that receives the replacement notification checks to see if it is affected (i.e. the lost device is a parent, sibling or child). If the device is affected then that device sends a message to the new $tier^{n}$ cluster-head in order to link with it. The device also forwards the replacement notification to any parents, siblings and children that are either $tier^{n+1}$, $tier^{n}$, or $tier^{n-1}$ cluster-heads where n represents the tier of the lost cluster-head.

3.4 NOA_{N-CH} and NOA_{1-CH} Variants

The selection algorithm used to determine the cluster-head of the next tier depends on whether the solution is a NOA_{N-CH} or NOA_{1-CH} variant (Fig. 3.2). The $N - CH$ variant allows a device to be a cluster-head at each tier. This means

that the $tier^n$ cluster-head can choose to become a $tier^{n+1}$ cluster-head which reduces the overhead associated with hierarchy construction and maintenance.

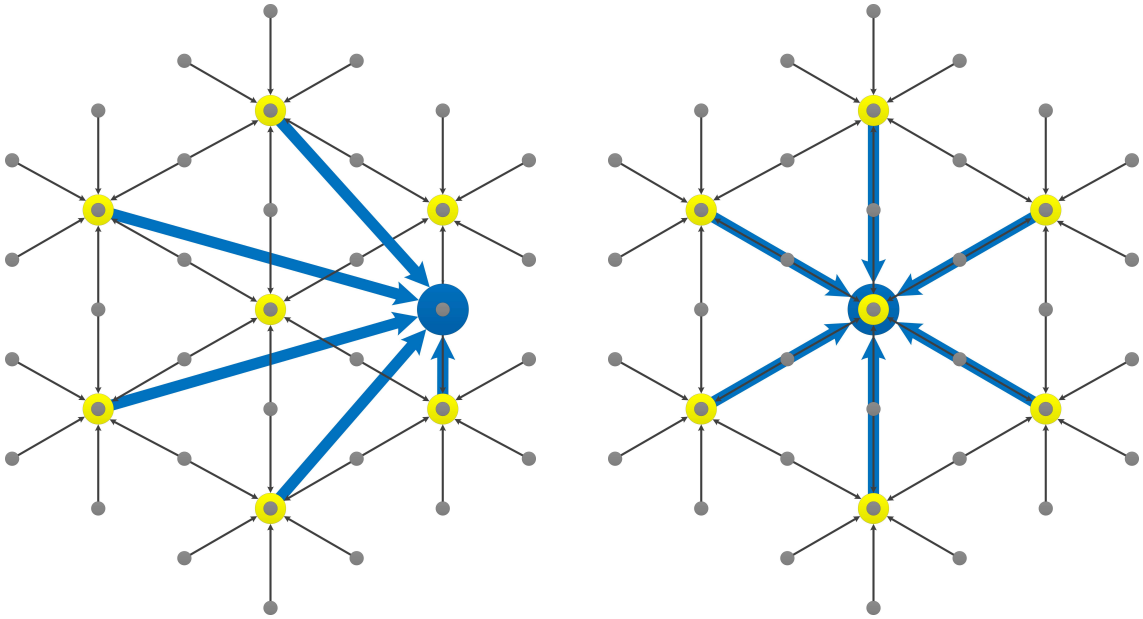
The $1-CH$ variant limits a device to acting as a single cluster-head. As such, a $tier^n$ cluster-head cannot also become a $tier^{n+1}$ cluster-head and instead must find a device in the local area that is not currently acting as a cluster-head. A randomized depth first search over the descendants of a cluster-head can be used to ensure that a device within the cluster's bounds is selected if such a device exists. This requires that a $tier^n$ cluster-head, where $n > 0$, randomly selects a child to nominate a cluster-head for it. If however the nominating cluster-head is a $tier^0$ device and it is not acting as a cluster-head the nominating cluster-head nominates itself and informs the original device that it will be the new cluster-head. If the $tier^0$ device is already acting as a cluster-head it sends a reply to its parent informing the parent that it is already busy. When the parent receives a busy reply it tries another randomly selected child from the children that have not already been selected. When a parent is informed that all of its children are busy it sends a reply to its parent signifying that there are no available children. If the original cluster-head is informed that all of its descendants are busy, the selection algorithm can opt to use an $N-CH$ type solution to ensure that the hierarchy remains intact on a depleted network.

The different constraints provide a trade-off between hierarchy properties. Devices that act as multiple cluster-heads are a single point of failure and have an

additional workload. However allowing a device to act as multiple cluster-heads reduces the configuration and aggregation overhead when compared to the $1-CH$ approach. This is because in the $N-CH$ solution a high-level cluster-head is usually also a number of sub-cluster-heads at lower tier levels, which allows the network to remove the overhead associated external communication when electing a parent as well as aggregating data. The $1-CH$ solutions require additional overhead in order to search for a suitable cluster-head to nominate (i.e. a device in a given proximity that is not currently acting as a cluster-head).

3.5 Cluster Synchronization

$Tier^1$ cluster-heads synchronize their children by periodically broadcasting a synchronization message. The children use the time value sent with the synchronization message to calculate an offset for that cluster. Once the offset is calculated for a cluster the device is synchronized with any other device in that cluster. For example, if devices A, B, and C are all in the same $tier^1$ cluster and C is the cluster-head. C broadcasts its current time, T_C, and devices A and B use T_C to offset the time for all cluster C events. As such, if A and B have scheduled a communication at time t then A and B need to wait $waitTime = t - T_C$ when $waitTime >= 0$, otherwise the devices have missed the event and need to handle the error appropriately.



(a) Two-Parent 1-CH (b) Two-Parent N-CH

Figure 3.2: Examples of the $1-CH$ and $N-CH$ two-parent clustering variants. The smallest gray circles represent $tier^0$ devices. The medium sized yellow circles are $tier^1$ cluster-heads. The largest blue circle represents a $tier^2$ cluster-head. In the N-CH version the $tier^2$ cluster-head is also a $tier^1$ cluster-head, but this is not the case in the $1-CH$ variant. Adding this constraint requires more overhead but removes a single points of failure and reduces the recovery cost if a high tiered cluster-head is lost.

CHAPTER 4. SIMULATION ENVIRONMENT

This chapter overviews the simulation design and the simulations that were performed. The simulation design is detailed in §4.1. A simulation run is defined in §4.2. An overview of the various simulations that were performed can be found in §4.3.

4.1 Simulator Design

The mesh package in ns-3 NS-3 [2009], which includes an 802.11s physical layer, was modified and extended to build a simulation test-bed for the NOA and HB protocols. NOA and HB were both implemented as application-level protocols. The simulator allows the user to specify the number of parents, whether a hierarchy is $1-CH$ or $N-CH$, and if local synchronization for duty-cycled devices is required.

4.1.1 Simulator Selection

Simulator selection is a difficult problem for wireless sensor networks. OMNet++/OMNest Varga and Hornig [2008], SimPy Muller and Vignaux [2003], OPNet Chang [1999], ns-2 McCanne et al. [1997], and ns-3 NS-3 [2009] are a few examples of

the simulators that can be used. Selection of a simulator can be based on its license, its ease of use, the available modules/protocols, memory consumption, computational performance or any combination of these and/or other considerations. In this case ns-3 was selected for a combination of the these considerations. It is licensed under the GNU GPLv2 license License [1999], making ns-3 publicly available for research, development and use. At the time of simulator selection the 802.15.4 MAC protocol IEEE [2003] was under development and was planned to be released soon. In the event that 802.15.4 was not released as scheduled, which was the case, there was the ns-3 mesh package that could be used after modifications for the simulations. The final deciding point was ns-3's scalable memory consumption and computational time, as well as its ability to simulate large scale networks consisting of thousands of devices without a significant increase in memory or time Weingärtner et al. [2009].

4.1.2 MAC/Routing Protocols

The 802.11s physical layer was utilized to implement cross layer MAC/routing protocols to support the differences in the requirements of NOA and HB. Each protocol was designed such that it could be used with and without synchronization support for duty-cycled devices. The first protocol MAC/routing protocol, Sync_NOA, is utilized by the NOA algorithm and does not allow flooding. In a duty-cycled network,

Sync_NOA uses the $tier^1$ cluster-heads to perform the local synchronization of devices in their clusters by periodically broadcasting a synchronization message. The second MAC/routing protocol, Sync_Flooding, provides support for flooding the network a specific number of hops. In order to support flooding in a sparse distribution with duty-cycled devices, every device periodically broadcasts a synchronization message.

When sending a uni-cast message in either NOA or HB the routing algorithm uses hierarchical routing where each cluster-head keeps a routing table that includes paths for each parent, child and sibling. NOA and HB perform these updates during hierarchy maintenance. Parent-to-child links are updated using the normal hierarchy maintenance and the descendant to ancestor routing information is updated with the heartbeat messages. HB uses its hierarchy creation algorithm to create and maintain the descendant-to-ancestor routing information. Ancestor-to-descendant routing information is created by having children acknowledge the receipt of the flooded message.

The flooding algorithm implemented in the simulator for HB allows a device to specify the number of hops a message should be flooded. Each flooded message includes a sequence number which is used to reduce the number of re-transmitted messages by allowing a device to only forward a message once. A device keeps track of the original sender's last received sequence number. It compares the stored sequence number with the latest sequence number and if the latest sequence number is greater it

re-broadcasts the message. This simple sequence number check requires that flooded messages are received in the order that they are sent. As this is not always the case, early termination along a forwarding path is possible, which can cause problems when constructing and maintaining hierarchies. Requiring a delay between successive messages flooded by the same device can help solve some of these time dependent issues.

4.1.3 Device Distribution

NS-3's mobility package was used to distribute devices on a rectangular network area. The grid position allocator was extended to distribute devices in a hexagonal grid similar to the ideal distribution (Chapter 2), but it distributes devices over a rectangular area as opposed to the ideal hexagonal area. The uniform random distribution is used to distribute devices such that the device density is similar across the network. The uniform distribution was set to provide every non-edge device with approximately six neighboring devices, where devices on the network's edge have a similar device density but fewer neighbors (Figs. 4.1 and 4.2).

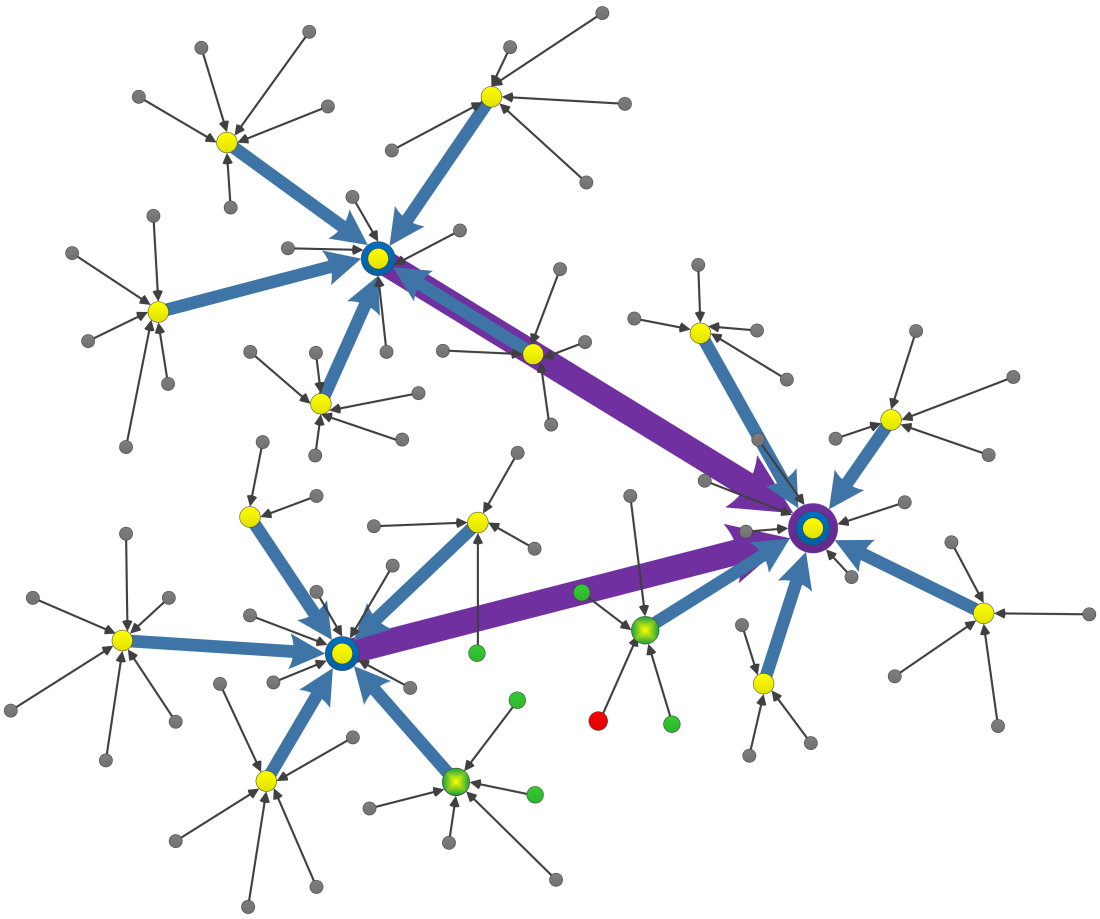


Figure 4.1: This example illustrates the architecture of the single-parent hierarchy on a random distribution of devices. The arrows represent a logical multi-hop link between a child and its parent. The devices in red and green show the network division problem. Notice how the red device can only communicate with the green neighbors that are part of its hierarchy and how the data from the red device and the neighbors that are not in its cluster is not compared until a common ancestor (i.e. the purple cluster-head).

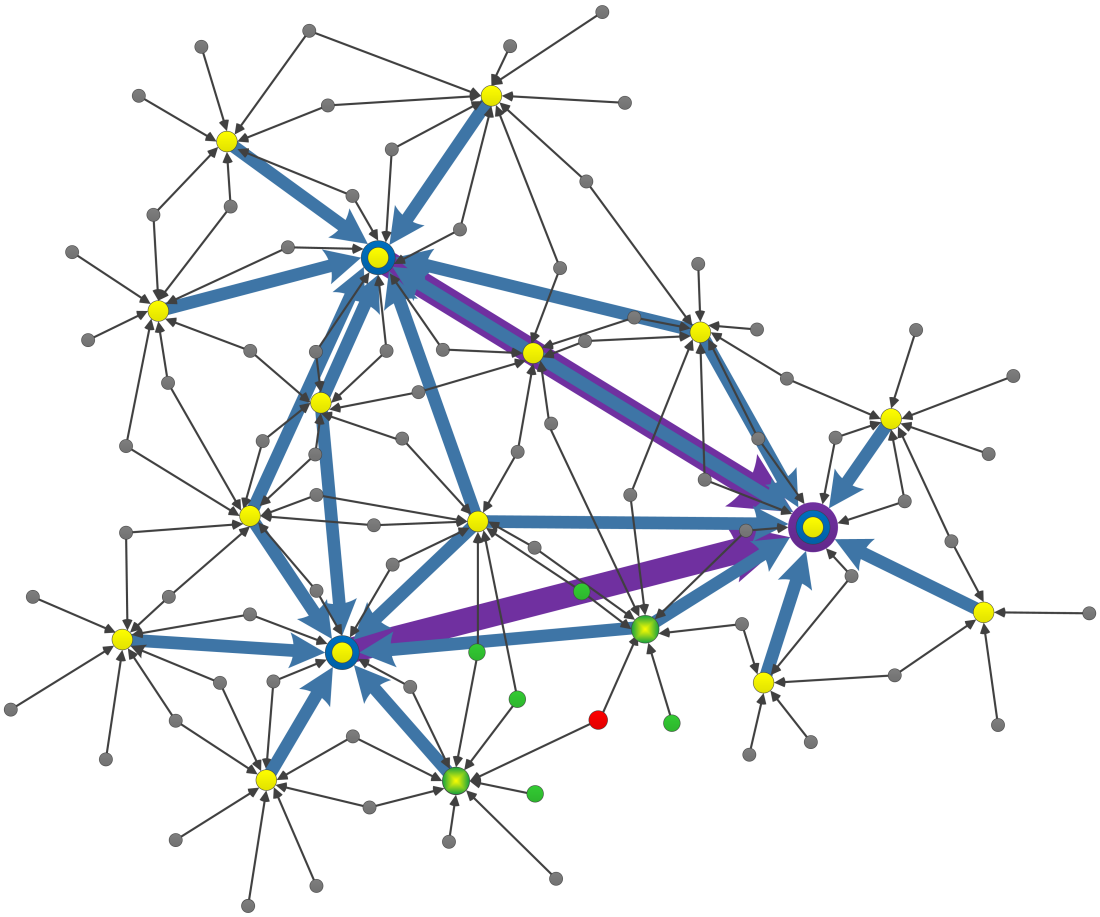


Figure 4.2: This example illustrates how the two-parent hierarchy is constructed on a random distribution of devices. The arrows represent a logical multi-hop link between a child and its parent. The devices in red and green show how the multi-parent hierarchy solves the network division problem seen in single-parent hierarchies. Notice how in the two-parent hierarchy the red device is in at least one cluster with all of its green neighbors.

4.1.4 Cluster-head Selection

Cluster-head selection is performed by virtually overlaying a hexagonal grid on the devices and having the device closest to the center of each hexagon act as the main device for that specific hexagon. Once each device determines which hexagon it represents the cluster-head selection algorithms outlined in §3.2 are used. This ensures that similar hierarchies are constructed across each algorithm to provide a fair comparison of the different protocols without having to worry about the idiosyncrasies of different hierarchies created by randomly choosing cluster-heads. In order to introduce a variety of hierarchical structures and change in the cluster-head placement across different runs, the initial x and y values used in the cluster-head selection algorithms are offset. This process shifts the location of the cluster-heads and can change the number of cluster-heads in each tier due to edge cases.

The hexagonal overlay helps to ensure that the portion of the hierarchy that was defined by this protocol is as similar as possible. However, the cluster-selection algorithm, §3.2, does not define cluster-heads for the edge/corner cases that are required for a rectangular device distribution. As such, cluster-heads that are required for the special cases are randomly selected and assigned. Further, the communication protocols can also create variances in the hierarchies created by the different protocols.

4.2 A Simulation Run

The concept of a simulation run is used to increase comparability between the different protocols. In a given run the seed values for equivalent random number generators are set to the same value. For example, the random number generators used in distributing the devices are set to the same seed numbers. This helps ensure that there is an equivalent network distribution for networks that are the same size, and that they are distributed over an area of the same size, no matter which other parameters are varied.

A simulation run generates a set of initial values for the simulator and consistently sets as many values as possible to these initial values. Only values that are required to differentiate two different simulations are changed. Then the simulators are started and run to completion. Beyond the initial values being equivalent it is important to try as much as possible to ensure that any values that significantly affect the simulations are equivalent at each point in the simulation. This issue can be addressed in part by ensuring that a random number generator is only used for one specific function (i.e. use a random number generator for device distribution and a separate one for cluster-head selection).

While using multiple random number generators helps, it is still possible that the differences in the simulations affect when random numbers are generated and as

such the overall outcome of the simulation. Take for example the differences between flooding a message and sending a uni-cast message. If a random number generator is used to determine the forwarding delay at a specific device then a flooded message can cause a device to generate a delay before re-broadcasting, but that device might not be used when forwarding a uni-cast message. As such, the random number generated to determine forwarding delay at that device could be different for the next message it forwards in the different simulations.

The stop time and simulation duration can significantly affect the comparability between two different simulations. NS-3 was designed such that the user specify the duration of the simulation (i.e. the number of simulated seconds from when the simulator starts to when it stops). This is useful when comparing throughput, bandwidth, etc. across different simulations as the time needs to be strictly defined. However, when running a simulation on hierarchy creation this can cause problems as it can take longer for one protocol to create a hierarchy than others, or to create a large network compared to a smaller network. If the simulation time is not long enough to construct the hierarchy that takes the most time, then when the simulation ends the hierarchies will be in different states of completion which makes it difficult to directly compare the results. If, however, a hierarchy is created well before the simulation ends then overhead messaging of the underlying protocols can skew data. Thus, to solve the problem NS-3's simulation environment was extended such that a user can

specify when a simulation ends based on specific network/hierarchy properties (e.g. when a hierarchy is created, after a specified number/percent of devices depleted their power reserves).

4.3 Simulations

The simulator was used to gather data on hierarchy construction cost, data aggregation cost and the lifetime of a network, including hierarchy construction, maintenance and data aggregation. Simulations were performed on rectangular networks of 200 (10x20), 800 (20x40), 1800 (30x60), and 3200 (40x80) devices.

Construction cost simulations were performed with non-duty-cycled devices and the hierarchy maintenance and data aggregation functions disabled. This allows the construction cost simulations to focus solely on the cost of constructing the hierarchy. To ensure that the same number of tiers was generated for each simulation the cluster-head selection algorithm placed the highest tiered cluster-head in the same corner for each hierarchy. In order to ensure the same distribution of devices across the different sized networks the devices were distributed in a hexagonal grid. The simulations were configured to end once the highest tiered cluster-head was configured.

The data aggregation simulations involved sending a fixed length value from a child to its parent. Each parent sent a value of the same size to its parents. In order to

ensure that the message length between the higher tiered cluster-heads was equivalent to that of the lower cluster-heads, each parent calculated an average over all of the values it received before forwarding the data to its parents. These simulations were performed on all of the different network sizes with both the *every parent* and *round robin* aggregation algorithms. The simulations utilized non-duty-cycled devices such that any overhead associated with synchronization could be ignored. They started with the hierarchy construction phase ignoring any cost associated with this phase. Once the hierarchy was constructed, the simulator sent data once every round and recorded the number of hops required to aggregate a value from every device to the top most cluster-heads for each round. Aggregation continued until at least 75% of the devices depleted their energy reserves, at which point the average aggregation cost per round of aggregation was calculated.

The lifetime simulations involved distributing the devices, selecting the cluster-heads, local synchronization, configuring and maintaining the hierarchy and data aggregation. The simulations ended when at least 75% of the devices depleted their energy reserves. Devices were distributed using the uniform, random placement algorithm. Duty-cycled devices were utilized, and once a device was discovered by the hierarchy it placed its radio in a low power state except during scheduled periods of time for communications purposes. The N-CH type hierarchy along with a round robin style data aggregation were selected for these simulations.

The lifetime simulations were performed to investigate the effects of varying the number of devices in the network, the synchronization period and the aggregation and maintenance period. A 5-minute synchronization and aggregation/maintenance period was used when varying the number of devices in the network. Three sets of simulations were performed on eight-hundred node networks while changing the synchronization and aggregation/maintenance period. The first set involved a fixed 5-minute synchronization period, the second fixed the aggregation/maintenance period to 5-minutes and the third varied both periods such that they were equivalent for each run. The periods that were not fixed were set to 1-, 5-, 10-, 15- and 30-minute periods.

CHAPTER 5. EVALUATION AND ANALYSIS

This chapter presents an evaluation and analysis of the various hierarchies, cost models and simulations. Section §5.1 discusses various aspects of the single-parent and multi-parent hierarchies from a qualitative view. The simulated and modeled hierarchy construction cost for the various hierarchies and algorithms is presented and compared in §5.2. Likewise §5.3 presents the aggregation cost for the various hierarchies and algorithms. A comparison of the network lifetime for the hierarchies and algorithms with regards to the number of nodes in the network, the synchronization period as well as the data aggregation and maintenance period is discussed in §5.4. A summary of the results is presented in §5.5.

5.1 Discussion of Single- and Multi-Parent Hierarchies

The single-parent hierarchy can be thought of as a hierarchy of logical star networks where each sub-cluster is a disjoint set of nodes with a single cluster-head. This structure creates an artificial division of the network that reduces the connectivity of the structure and creates a possible single point of failure at each cluster-head (Fig. 4.1). In the single parent cases a network division is created by the fact that sub-clusters are disjoint sets and as such the severity of the division between neighboring

clusters depends on the clustering at the higher tiered levels. If a division was generated by the sub-clusters of the highest tier, $tier^{N-1}$, then the structure divides the network at the highest level and communication between neighboring clusters could have to pass through the highest tiered, $tier^{N}$, cluster-head. Considering the other extreme, a network division created by $tier^{1}$ clusters that are part of the same $tier^{2}$ cluster can be circumvented by communicating through the $tier^{2}$ cluster-head.

These artificial structural divisions can increase the complexity of an algorithm that uses the logical structure as a paradigm. For example, an in-network, multi-resolution analysis algorithm that performs data aggregation and analysis at each cluster-head would be subject to the network divisions. Thus, the comparison between data from the two clusters/devices would only be performed at the common ancestor. This means that the data from two neighboring devices might not be compared until the highest tiered cluster-head and until the data has been aggregated, compressed, etc. such that the data does not fully represent either of the two individual devices.

The multi-parent recursive area hierarchy can be thought of as a hybrid of a hierarchy of logical star networks combined with a logical mesh network. The star network hierarchy provides the scalability of the solutions, while the mesh network provides a logical connection between neighboring clusters at each tier. This hybrid structure can remove network divisions, increase network robustness and redundancy, and provide significant benefits for data aggregation and analysis routines.

The multi-parent hierarchy connects neighboring clusters by overlapping each cluster such that neighboring $tier^n$ cluster-heads have at least one common child at $tier^{n-1}$. The utilization of multiple parents for each cluster adds redundancy to in-network data aggregation, analysis and storage. The redundancy increases the network robustness as well as the robustness of the algorithms that utilize the multi-parent paradigm. Performing analysis at each cluster-head provides an in-network, multi-resolution analysis where data from neighboring data generation devices is compared at a common grandparent (i.e. two tiers above the data generation). It is important to note that a data generation device can be a device that is gathering raw data, or a cluster-head that is aggregating data. Further, the data from a generation point can be compared with slightly different sets of devices at each parent. This can be used to increase the reliability of anomaly detection routines as the data from a device can be compared with different sets of surrounding devices at each parent and then with all of the surrounding devices at the grandparent. As such, if a phenomenon change occurs at the edge of a cluster the analysis routine could flag it as anomalous to one cluster but not another. In this case the next level of analysis can determine if the data representing the possible anomaly fits an acceptable phenomenon pattern in the local area. If however the data was only reported to one cluster, it is possible that the data could incorrectly be marked as an anomaly, or as a normal case, depending on which cluster the device reported the data to.

Agent based distributed analysis routines can also benefit from the multi-parent structure. For example, an agent based object tracking algorithm can track an object as it moves through the network Tseng et al. [2003]. An agent, an entity that represents data about an object, is created when an object is first detected. The agent then follows the object through the network by being passed from one device to the next as the object moves through the network. Information about the object's current position and path are sent to a base station.

The logical multi-parent recursive area hierarchy structure provides agents with communication paths between clusters while also providing a scalable reporting mechanism to data sinks. The communication paths between clusters is beneficial as agents can exist at any tier in the network, and neighboring clusters can inform the agent which device it should move to next and at what time it should move. The hierarchy can also be used to reduce the cost of reporting an object's position. Assuming multi-resolution object tracking where a cluster-head can query nodes in its cluster for additional information, a cluster-head only needs to know when an object enters or leaves its coverage area. This is because the cluster-head only needs to know of the object in its area of coverage and if it needs to know the current location of said object it can query its children to drill down to a more detailed object location. Further, the overlap between neighboring clusters provides an agent with communication paths to any cluster-heads that are affected as the object moves through the network.

The redundancies included in the multi-parent hierarchy remove the single point of failure that each cluster-head represents in a single-parent hierarchy. This redundancy can increase hierarchy robustness as any non-edge device has at least one back-up cluster-head. Further, when data is stored at each cluster-head there is redundancy in the data storage so if a cluster-head is removed from the network the data stored on that device can be recovered from the sibling cluster-heads as well as the parents and children.

5.2 Hierarchy Construction

The cost, in terms of number of message hops, to construct the initial hierarchy for a network that is distributed in a hexagonal grid depends on: the protocol, the number of parents, the number of nodes in the network, the shape of the network and the placement of cluster-heads. The network model assumes that every cluster in the hierarchy is complete and sets up the network such that it is distributed in a hexagonal shape with a single $tier^N$ cluster-head placed in the center of the network. The simulator allows partial clusters to exist and sets up the network such that the nodes are distributed over a rectangular area with a single $tier^N$ cluster-head in the hexagonal grid at position (0,0).

Figure 5.1 and table 5.1 show the simulated construction cost for $N-CH$ type

Table 5.1: The cost, in terms of number of message hops, required to configure N-CH type hierarchies using the different protocols on networks with various sizes.

Number of Nodes	200	800	1800	3200
HB_1	4.45E+02	2.74E+03	7.33E+03	1.26E+04
HB_2	1.05E+03	5.54E+03	1.21E+04	2.84E+04
HB_3	1.92E+03	1.07E+04	3.34E+04	5.71E+04
NOA_2	8.27E+02	3.10E+03	7.40E+03	1.36E+04
NOA_3	1.55E+03	6.70E+03	1.33E+04	2.51E+04

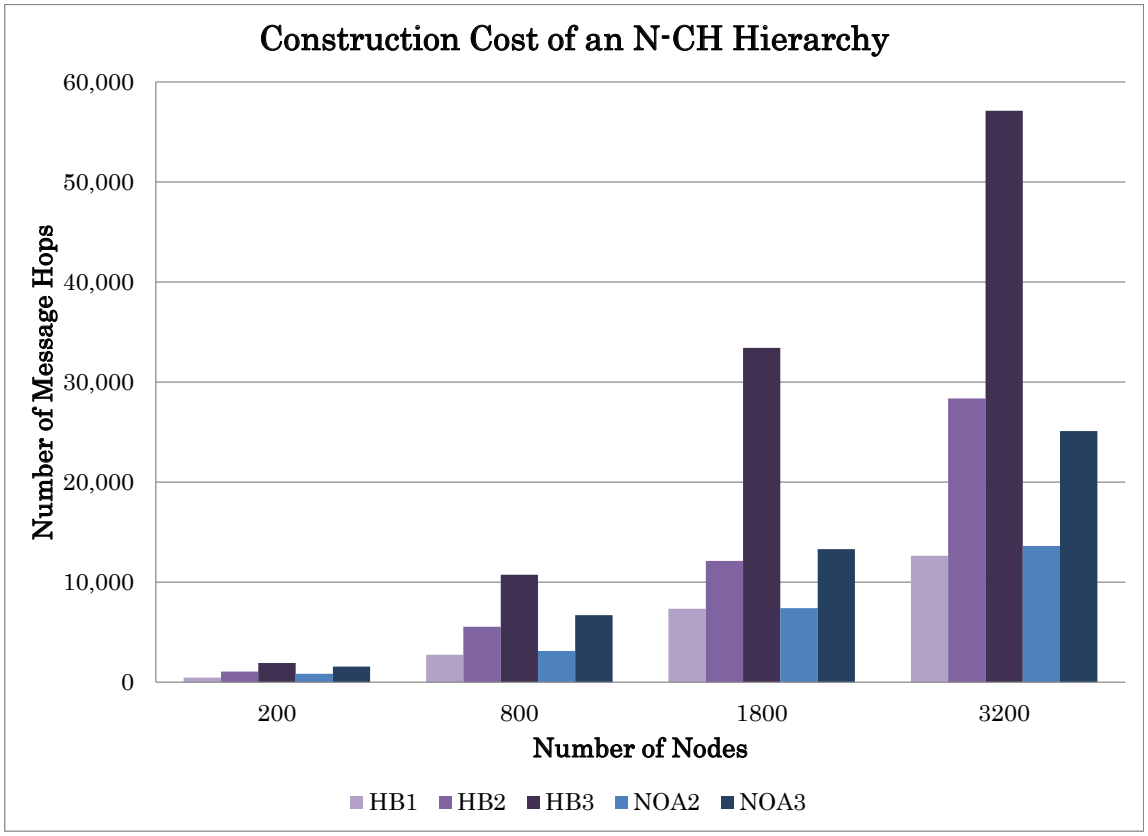


Figure 5.1: The cost, in terms of number of message hops, required to configure N-CH type hierarchies using the different protocols on networks with various sizes.

networks with different numbers of nodes. In all cases NOA reduces the construction cost of an equivalent hierarchy when compared to hierarchical beaconing (HB). Configuration of a hierarchy using NOA_2 can be performed using between 21% and 52% fewer message hops when compared to HB_2. Similarly NOA_3 requires between 20% and 60% fewer message hops compared to HB_3. In both cases the larger the network the more significant the construction cost reduction is. In the networks with 800 or more nodes, NOA_2 is able to configure a network with only 1% to 13% more message hops than HB_1, providing the hierarchy with the benefits of a multi-parent hierarchy at a similar number of message hops for construction. While NOA_3 requires 79% to 116% more message hops than NOA_2, the three-parent hierarchy provides an additional cluster-head that can be utilized by voting algorithms when performing in-network analysis.

The $1-CH$ type protocol restricts a device such that it can only act as a single cluster-head at any given tier. As such $1-CH$ construction has the additional cost of finding a suitable descendant (i.e. a descendant that is currently not acting as a cluster-head) when compared to $N-CH$ type hierarchies which allow a device to act as a single cluster-head per tier. The additional step of finding a suitable descendant accounts for part of the difference in hierarchy construction cost that is shown in Figure 5.2. The $1-CH$ type hierarchies have to also account for the increased maximum distance between a cluster-head and the cluster's edge or cluster-

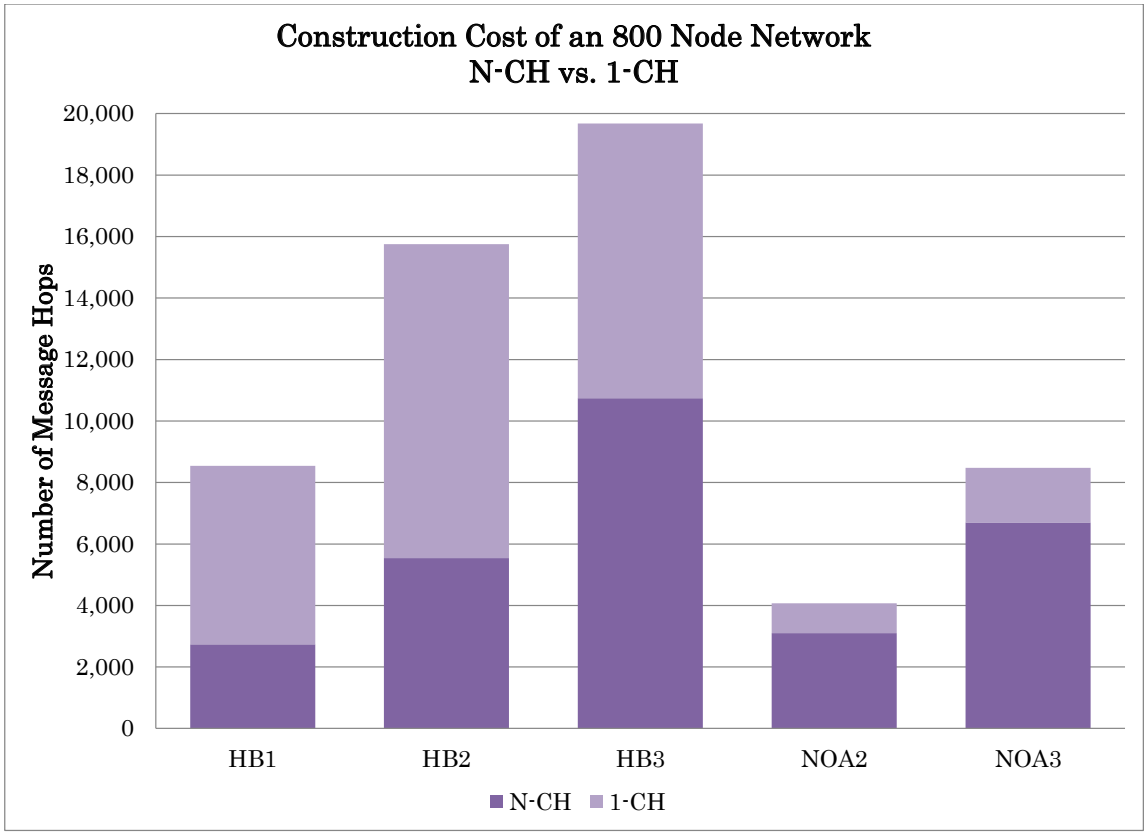


Figure 5.2: The simulated configuration cost of an 800 node network configured using the $N-CH$ protocols is compared with equivalent networks configured using the $1-CH$ protocols.

head's children. This is because with the $N - CH$ type hierarchy a cluster-head is centrally located in the cluster, however with the $1-CH$ type hierarchy a node on the cluster's edge could be selected as the cluster-head. Thus $1 - CH$ type hierarchical beaconing solutions need to double the flooding radius in order to ensure that all of the devices in a cluster receive a flooded message. NOA's increased cost due to the random placement of a cluster-head within its cluster is less extreme as NOA only has to send a uni-cast message approximately four times farther for the worst case. The benefits of $1 - CH$ type hierarchies are: a reduction in single points of failure (i.e. no cluster-head acts s a cluster-head or more than one tier), reduction in the cost of replacing a cluster-head that is used by multiple tiers and distribution of the work load associated with being a cluster-head at multiple tiers to multiple devices which can help balance the power consumption, routing overhead, and computational workload.

A comparison between the network model and the simulation data is presented in Figure 5.3. In order to compare the simulation data and the model two adjustments were made to the network model to account for the basic model using a maximum tier to provide the cost estimations as opposed to a number of nodes. For example, the basic model estimates an equivalent cost for a 344 node and a 2401 node network and as such overestimates the construction cost of the 344 node network. The first adjustment to the model is to account for the possible difference in the number of

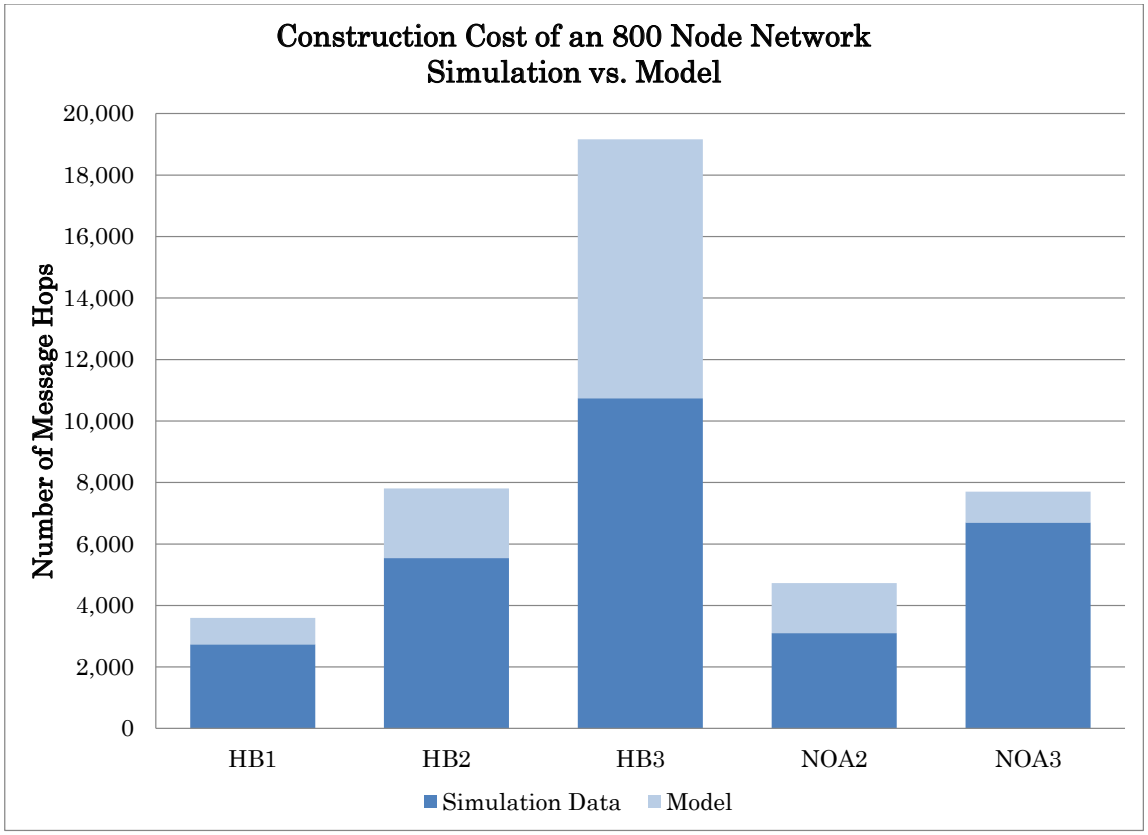


Figure 5.3: Comparison of the estimated configuration cost using the adjusted network model with the simulated configuration cost. The model overestimates the cost as it does not account for all of the instances of partial clusters.

nodes in a $tier^N$ network. It is performed by finding the ratio between the actual network size and the maximum supported network for the required tier (in the example it would be 344/2401). This ratio is then multiplied by the estimated cost (i.e. $344/2401 * 777$).

The second adjustment is used to address the difference in the radius of a network that is smaller than the maximum radius of the $tier^N$ cluster. It also accounts for the change in the radius due to the placement of the highest-tiered cluster-head and the shape of the distribution. Referring to the previous example, the maximum radius of a single-parent $tier^4$ cluster is 40 hops. However, a 344 node network has a radius closer to 14 hops if configured using a hexagon. The radius is around 18 hops if the network is distributed in a 9 x 18 rectangle and the highest-tiered cluster-head is in a corner of the network, as is the case in the construction cost simulations. This adjustment changes the estimated cost to configure a cluster-head (Formulas 2.17, 2.20, 2.23, 2.26 and 2.29) by changing the $RadiusOfCluster(t)$ function such that it returns the network's hop diameter, if the network diameter is less than the cluster-radius otherwise it returns the cluster-radius. With regards to the shape of the network and placement of the highest tiered cluster-head, this adjustment only accounts for the change in the network's radius and ignores any other effects of these differences. Further, it only applies to the highest-tiered cluster, thus the configuration cost for lower-tiered partial clusters remains unaffected.

These adjustments to the model provide a better fit when applying the model to the simulated situation but do not account for all of the differences between the model and the simulation. As such, the model still overestimates the configuration cost. In general the model configuration cost for hierarchical beaconing (HB) is influenced more severely by these differences because the distance that a message is flooded exponentially impacts the total number of message hops. A uni-cast message however is one-to-one, where increasing the distance a message travels has the same impact on the required number of message hops when ignoring the overhead associated with dropped/corrupted messages (Fig. 5.3).

The number of parents also affects the accuracy of the model. Increasing the number of parents for NOA increases the accuracy of the model due to the fact that a three-parent hierarchy has clusters that are closer together, thus reducing the effects of the partial-cluster edge cases. However, increasing the number of parents when using HB increases the error because the errors associated with incorrectly estimating the cost of flooding a partial cluster outweigh the effects that an increased number of parents has on the severity of the partial-cluster edge cases. Further, since clusters overlap in the multi-parent hierarchy, error in the estimated cost for a cluster is duplicated in the two-parent hierarchy and triplicated in the three-parent hierarchy (Fig. 5.3).

Table 5.2: The cost, in terms of number of message hops, required to aggregate data using the various N-CH type hierarchies and network sizes.

Number of Nodes	200	800	1800	3200
HB_1	315	1268	2799	
HB_2	372	1504	3419	
HB_3	457	1935	4317	
NOA_2	372	1504	3416	
NOA_3	457	1935	4319	

5.3 Data Aggregation

The simulations of the protocols utilize the maintenance heartbeat messaging to forward data up the hierarchy. As such, the aggregation cost presented in table 5.2 should only to be considered when data aggregation is performed more often than maintenance. The presented data represents a round-robin style of data aggregation performed on an $N - CH$ hierarchy. Increasing the number of parents increases the cost of round robin data aggregation by 19% for the two parent hierarchy compared to the single-parent. The three-parent hierarchy requires 24% more hops than the two-parent and 49% more hops than the single-parent.

The same adjustments that are made to the configuration model need to be applied to the aggregation model. This allows the model to account for the differences between itself and the simulator and provides a more accurate comparison (Fig. 5.4). It is important to keep in mind that the adjustments still do not account for all of the

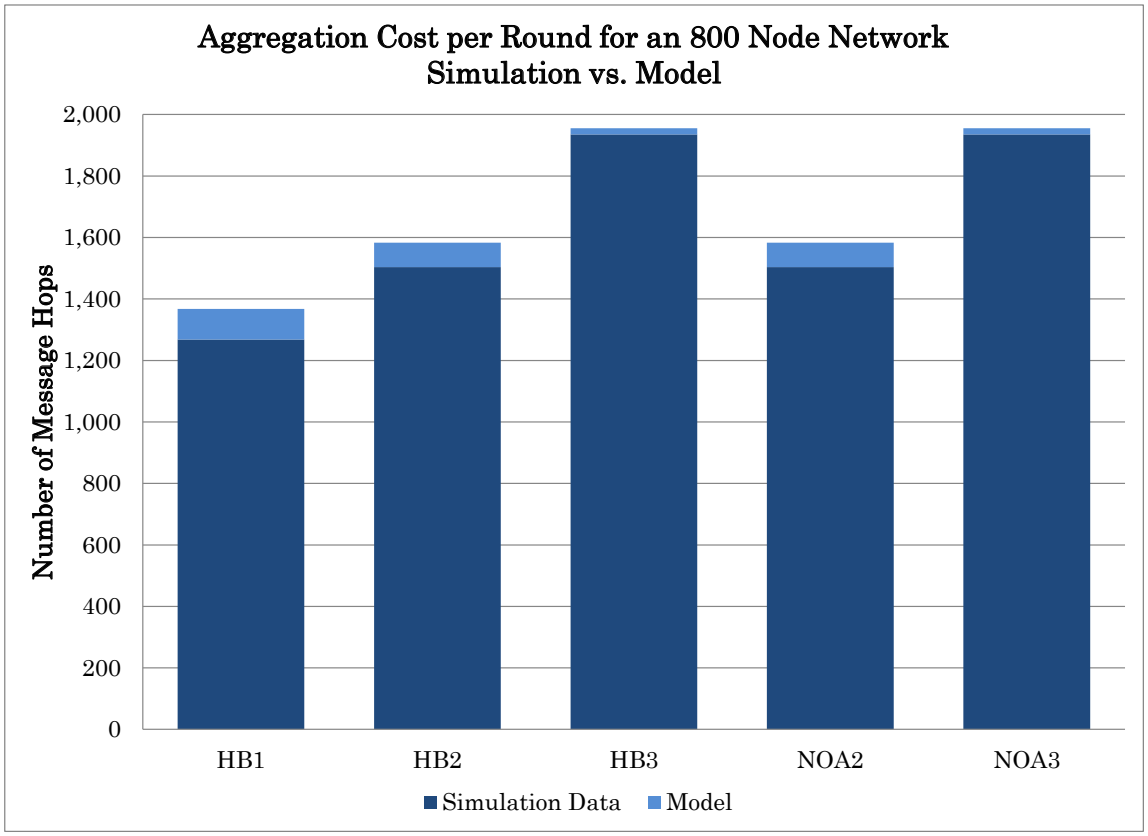


Figure 5.4: Comparison of the estimated per round aggregation cost using the adjusted network model with the simulated per round aggregation cost. The model overestimates the cost as it does not account for all of the instances of partial clusters.

edge cases and as such the model will overestimate the cost. Increasing the number of tiers in the hierarchy will tend to increase the error because the overestimation is compounded with each additional tier. However, increasing the number of parents in the hierarchy will tend to reduce the error since the additional parents reduce the significance of the edge cases. This is because increasing the number of parents increases the number of nodes that are overlapped between neighboring clusters and as such decreases the cluster size for clusters of $tier^2$ and higher.

5.4 Network Lifetime

The network lifetime is measured in minutes from network distribution and initial randomized start-up of the devices over a 15-minute period until at least 75% of the devices deplete their energy reserves. The following analysis of the network lifetime considers the protocol, the number of nodes in a network (Fig. 5.5), the local synchronization period (Fig. 5.7 and 5.6) and the data aggregation/maintenance period (Fig. 5.8 and 5.6). Data aggregation is performed with the maintenance messaging and as such these two periods are tied together. In all cases NOA_2 and NOA_3 extend the lifetime of the equivalent networks when compared to HB_1, HB_2, and HB_3. When compared with each other there is little or no difference between the lifetime of equivalent networks configured using NOA and when there is a difference

Table 5.3: The number of tiers that were configured for the various lifetime simulations.

Number of Nodes	200	800	1800	3200
One-Parent	3	4	4	4
Two-Parents	4	5	5	5
Three-Parents	4	5	5	5

NOA_2 provides a longer lifetime than NOA_3.

The number of nodes in a network affects the lifetime of the network because it affects the number of tiers required by the hierarchy. This is shown in Figure 5.5 and table 5.3 as the 800, 1800 and 3200 node lifetimes are similar but the 200 node network lifetime is longer. In this case the 800, 1800, and 3200 node networks were configured using 4 tiers for the single-parent hierarchy and 5 tiers for the two- and three-parent hierarchies. Even though the number of nodes change, the network lifetime remained similar. The 200 node network was configured using 3 tiers for the single-parent hierarchy and 4 tiers for the two- and three-parent hierarchies. As such the network lifetime is directly dependent on the number of tiers and indirectly dependent on the number of nodes. Further, the larger networks with the same number of tiers have an increased network lifetime due to the relative decrease in the number of edge cases; a smaller square has a higher perimeter to area ratio than a larger square. The other minimal differences between the 800, 1800, and 3200 node network are due to the intricacies of utilizing a pseudo-random simulation environment with different sized

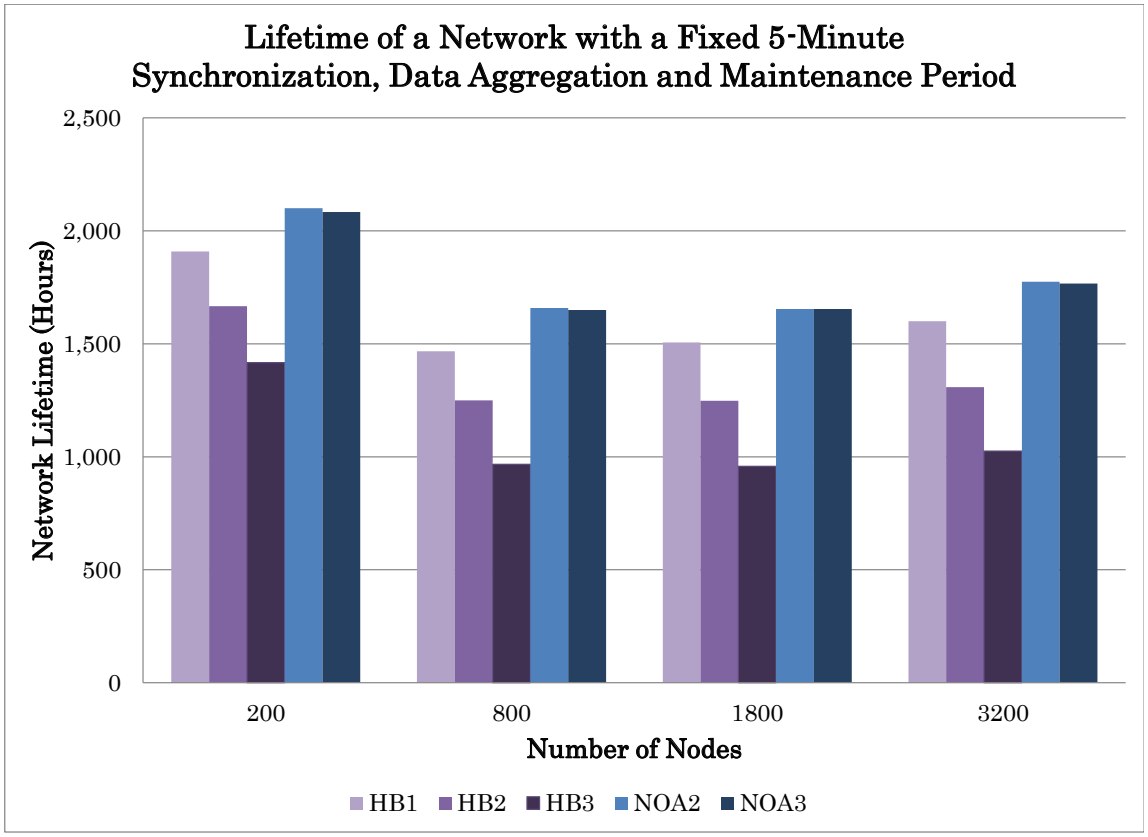


Figure 5.5: The change in the network lifetime considering networks with various numbers of nodes. This shows that the network lifetime and number of nodes are indirectly related through the number of tiers that the recursive area hierarchy requires for a given protocol and number of parents.

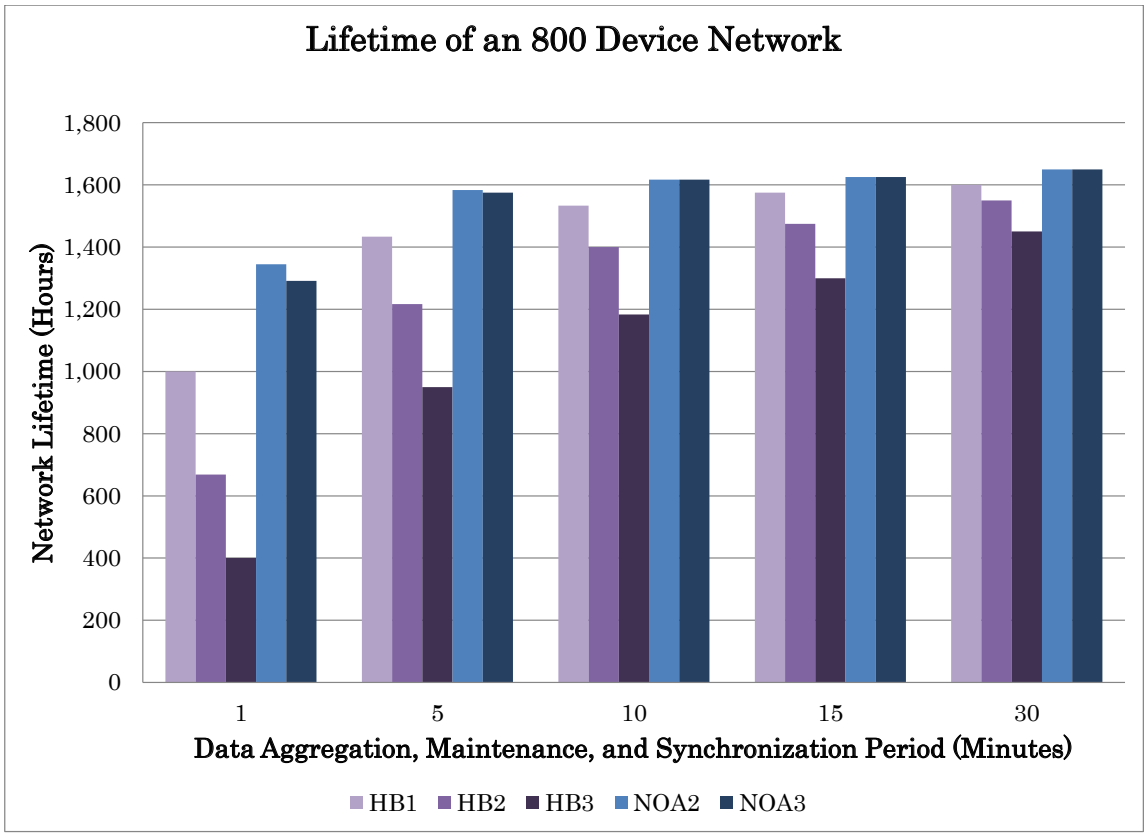


Figure 5.6: The change in the network lifetime considering various synchronization and data aggregation/maintenance periods such that for a given run both periods are equivalent.

networks as explained in 4.2.

Figure 5.6 illustrates the effect that varying the synchronization, aggregation and maintenance periods has on the network. In this case all three periods were set such that they were equivalent for a given run. The chart shows that a shorter period decreases the network lifetime. Considering the relative difference in the frequency of performing synchronization, aggregation and maintenance, a higher frequency has

a greater relative effect on the network lifetime. This increased effect is due to the fact that a duty-cycled device continues to consume power when in a low-power mode and the simulations account for this. While the relative difference is displayed more prominently in Figure 5.6 due to the difference in the ratio between adjacent categories. However, when the different ratios are accounted for there still exists a relative lifetime difference.

Figure 5.7 fixes the data aggregation/maintenance period to 5 minutes and varies the synchronization period, whereas Figure 5.8 fixes the synchronization period to 5 minutes and varies the data aggregated/maintenance period. Comparing the two figures shows that changing the data aggregation/maintenance period has more effect on the network than the synchronization period. Further, when considering the impact of a synchronization period >= 5 minutes (Fig. 5.7) there is little change in the network lifetime. This is because the fixed 5-minute data aggregation/maintenance period overpowers the difference between the various synchronization periods.

5.5 Summary of Results

In general NOA outperforms hierarchical beaconing for hierarchy construction cost when considering hierarchies with the same number of parents. When comparing NOA_2 with HB_1 the construction cost is roughly 1% to 10% higher for the larger

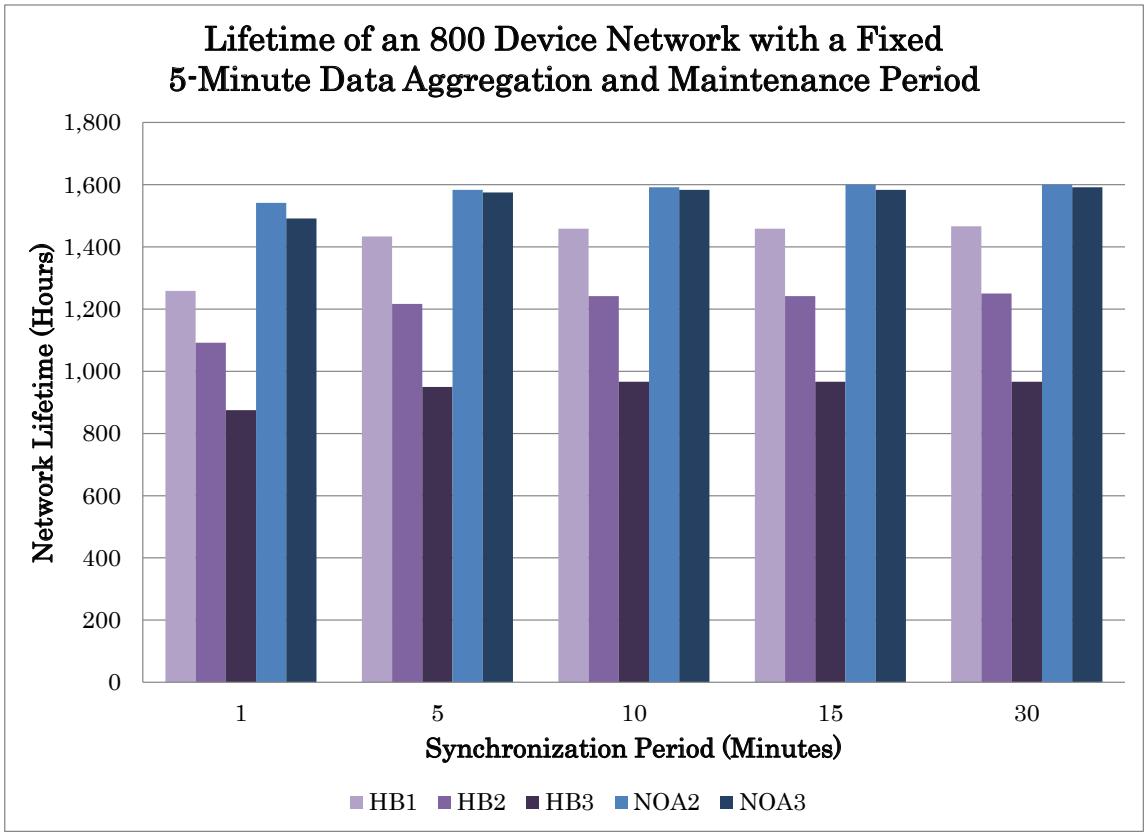


Figure 5.7: The change in the network lifetime considering a fixed data aggregation/maintenance period and various synchronization periods. The synchronization period has minimal effect on the network lifetime when compared to the data aggregation/maintenance period.

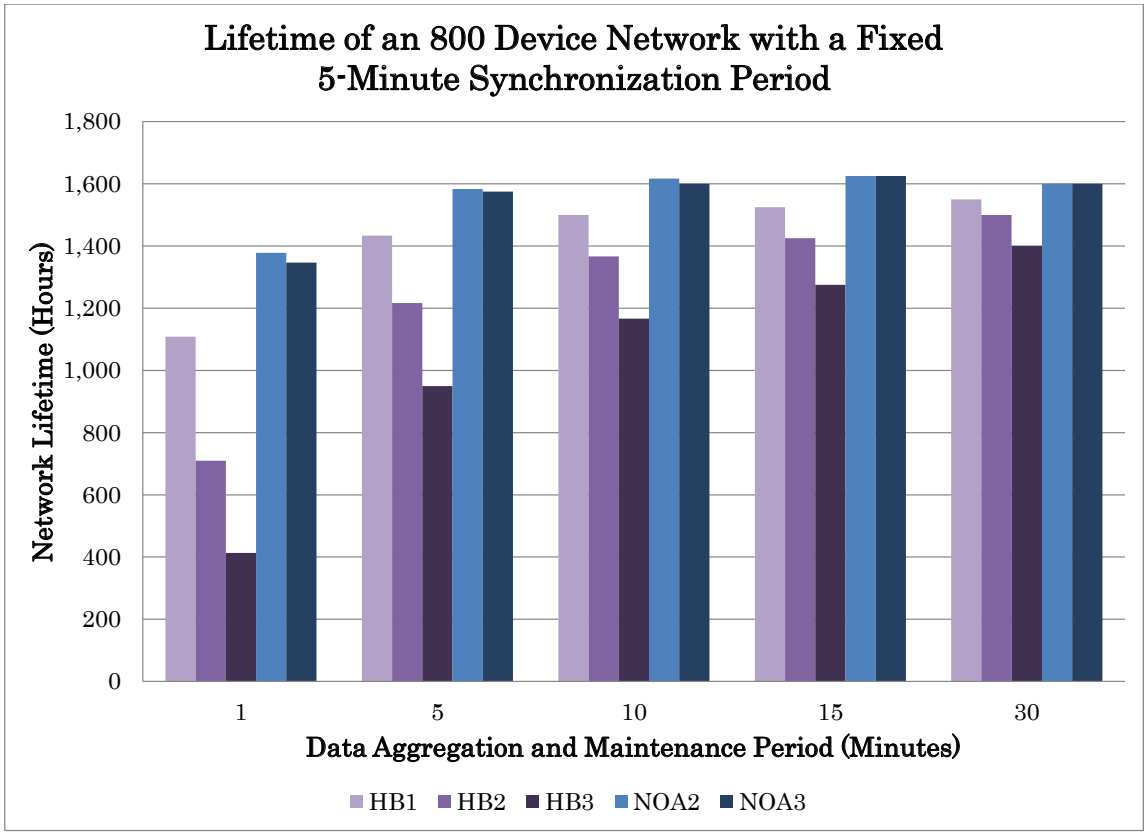


Figure 5.8: The change in the network lifetime considering a fixed synchronization period and various data aggregation/maintenance period.

networks. The 200 node network required 86% more hops, however the cluster-radius and other properties do not scale linearly and given a 200 node network these properties had not yet scaled beyond the two- and three-parent equivalents. $1-CH$ type hierarchies have an increased construction cost when compared to the $N-CH$ type hierarchies. This is due to the additional overhead associated with finding a suitable cluster-head and the increase in the average distance between a child and its parent. The network construction cost model over-estimates the construction cost for a hierarchy. The hierarchical beaconing is affected more severely due to its use of the flooding algorithm for construction. Hierarchies with more parents tend to see more error as well due to duplication of errors associated with partial clusters.

Data aggregation has a similar cost for both NOA and HB which is expected because the same aggregation protocol is utilized by each algorithm. Further, the similar cost for data aggregation between NOA and HB provides a level of validation showing that NOA and HB construct equivalent hierarchies. The data aggregation model provides a good fit with only minor over-estimation due to partial clusters.

A network configured using NOA can extend the lifetime of the network when compared to configuring the network with hierarchical beaconing using any number of parents. The number of tiers affects the network lifetime and since the network size has an effect on the number of tiers the network size indirectly affects the network lifetime. The data aggregation and maintenance period has a more significant

effect on the network lifetime than the synchronization period. Further increasing the synchronization, aggregation or maintenance period has a reduced effect because the power consumption of a device while sleeping over the extended time starts to outweigh the power used for the communications associated with synchronization, aggregation and maintenance.

CHAPTER 6. CONCLUSION

This dissertation presents a novel protocol for managing large-scale wireless sensor networks, NOA. Background information and related work are described in Chapter 1. The ideal network distribution and cost models are provided in Chapter 2. Details about NOA, the proposed protocol, are outlined in Chapter 3. Chapter 4 describes the simulation environment used to generate cost data about different protocols. An evaluation and analysis of the various hierarchies, cost models and simulations are discussed in Chapter 5. Contributions and future work are presented in this chapter.

6.1 Contributions

The contributions of this research include: NOA, a novel large-scale management protocol, construction and aggregation cost models, implementation and simulation of NOA, and an evaluation and comparison of NOA. The proposed protocol, NOA, is a novel protocol for autonomous configuration and maintenance of a multi-parent recursive area hierarchy for large scale wireless sensor networks. NOA is compared with single- and multi-parent variants of hierarchical beaconing. Construction and aggregation cost models are provided for the $N - CH$ type hierarchical

beaconing, gossip-based, and NOA protocols. $N-CH$ and $1-CH$ variants of NOA and hierarchical beaconing were implemented and simulated using ns-3. The simulation environment was used to compare the construction and aggregation cost of the different protocols, as well as the network lifetime. The simulations were designed to investigate the effects that the protocol, synchronization period, data aggregation/maintenance period, and network size have on the network lifetime.

- *The NOA protocols extend the lifetime of a network by 126% to 172% depending on the network size and protocol when compared to the corresponding multi-parent HB variants and 109% to 113% when compared to the single-parent hierarchical beaconing protocol. NOA's increased lifetime can be attributed to the reduced cost of hierarchy construction and maintenance as well as the reduced cost of synchronization.*

- *Decreasing the data aggregation, maintenance and synchronization frequency from once-a-minute to once-every-thirty-minutes increases the lifetime for a network configured with NOA by 123% for the two-parent case and 128% for the three-parent case.*

- *When compared to hierarchical beaconing, NOA can decrease the cost of hierarchy configuration by 52% for the two-parent hierarchy and 60% for the three-parent hierarchy. The relative difference in the cost of configuring a network*

using NOA compared to hierarchical beaconing does not scale linearly as the size of the network increases. As such, configuration of an exponentially larger network using NOA will typically result in a more significant decrease in cost of configuration.

- *When performing the aggregation of a single value from each device the hierarchies constructed by NOA exhibit an equivalent aggregation cost for the corresponding two- and three-parent hierarchies. The equivalent aggregation cost is expected and further validates that the hierarchies constructed using NOA are equivalent to those constructed using hierarchical beaconing. When comparing the single-parent hierarchy to the two- and three-parent hierarchies the two-parent hierarchy requires 16% more hops and the three-parent requires 43% more hops.*

When selecting an autonomous configuration and maintenance solution to maximize the lifetime of the network it is important to first look at the system requirements to determine which solution will be best, as there are various considerations that need to be accounted for before an appropriate protocol can be selected. The first consideration is whether the redundancies and other benefits of a multi-parent hierarchy are necessary for the application at hand. Another consideration is the frequency and level of synchronization that is required (i.e. every device synchronized with all of its neighbors vs. cluster-based synchronization provided by NOA). Other considerations

include: how dynamic the network will be, how quickly the management protocol needs to identify and react to a change in the network, how often data needs to be sent from each device to a collection point and if data reporting can be event driven as opposed to periodic. NOA can extend the lifetime of a network as well as provide the inherent benefits of a multi-parent hierarchy to a variety of large-scale wireless sensor networks capable of utilizing a multi-parent configuration.

This work is a starting point for researching the advantages, disadvantages and costs associated with utilizing multi-parent recursive area clustering hierarchies. Two main areas for future research include a broader investigation of the multi-parent structure for management purposes and utilization of the multi-parent structure as a paradigm for application level tasks, such as in-network data analysis.

The current research focuses on a sparsely distributed network of nodes with a static location. The broader investigation can look at the effects of changing the network density and could determine when and if a hybrid solution between the one-hop and multi-hop recursive clustering hierarchy is advantageous for dealing with an increased node density. Other research that could be included in a broader investigation would deal with mobile devices, including devices that have control of their mobility as well as devices that are carried.

6.2 Future Work

Future research that involves utilizing the multi-parent structure as a paradigm for application level tasks can take on many forms that require research into various applications. For example, performing agent-based object tracking using the multi-parent structure, as explained in §5.1, could provide a novel solution to the problem. An in-depth analysis of data aggregation and multi-resolution analysis that uses complex data from multiple sensors would extend the simplified data aggregation analysis that was performed for the current research. Further investigation into utilizing the multi-parent paradigm to perform multi-resolution anomaly detection and reporting could result in novel solutions that utilize redundant data analysis and a voting scheme to reduce error.

Bibliography

Ameer Ahmed Abbasi and Mohamed F. Younis. A survey on clustering algorithms for wireless sensor networks. *Computer Communications*, 30(14-15):2826–2841, 2007.

C. Alippi, R. Camplani, C. Galperti, and M. Roveri. A robust, adaptive, solar-powered WSN framework for aquatic environmental monitoring. *Sensors Journal, IEEE*, 11(1):45–55, 2011.

S.A. Aly, A. Ali-Eldin, and H.V. Poor. A distributed data collection algorithm for wireless sensor networks with persistent storage nodes. In *New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on*, pages 1–5. IEEE, 2011.

Panayiotis Andreou, Demetrios Zeinalipour-Yazti, Andreas Pamboris, Panos K. Chrysanthis, and George Samaras. Optimized query routing trees for wireless sensor networks. *Inf. Syst.*, 36(2):267–291, April 2011. ISSN 0306-4379. doi: 10.1016/j.is.2010.06.001.

Junaid Ansari, Dmitry Pankin, and Petri Mähönen. Radio-triggered Wake-ups with Addressing Capabilities for Extremely Low Power Sensor Network Applications. *IJWIN*, 16(3):118–130, 2009. URL http://dblp.uni-trier.de/db/journals/ijwin/ijwin16.html#AnsariPM09.

S. Bandyopadhyay and E.J. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1713–1723. IEEE, 2003.

S. Bhatti and J. Xu. Survey of target tracking protocols using wireless sensor network. In *Wireless and Mobile Communications, 2009. ICWMC'09. Fifth International Conference on*, pages 110–115. IEEE, 2009.

Olutayo Boyinbode, Hanh Le, Audrey Mbogho, Makoto Takizawa, and Ravi Poliah. A Survey on Clustering Algorithms for Wireless Sensor Networks. In Tomoya Enokido, Fatos Xhafa, Leonard Barolli, Makoto Takizawa, Minoru Uehara, and Arjan Durresi, editors, *NBiS*, pages 358–364. IEEE Computer Society, 2010.

Michael Buettner, Gary V. Yee, Eric Anderson, and Richard Han. X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In Andrew T. Campbell, Philippe Bonnet, and John S. Heidemann, editors, *SenSys*, pages 307–320. ACM, 2006. ISBN 1-59593-343-3.

W.R. Chang, H.T. Lin, and Z.Z. Cheng. CODA: a continuous object detection and tracking algorithm for wireless ad hoc sensor networks. In *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*, pages 168–174. IEEE, 2008.

Xinjie Chang. Network simulations with OPNET. In *Simulation Conference Proceedings, 1999 Winter*, volume 1, pages 307–314. IEEE, 1999.

Johnathan Vee Cree. Scalable Multi-Parent N-Tiered Clustering in WSNs. In *The 11th ACM/IEEE Conference on Information Processing in Sensor Networks PhD Forum*, April, 15 2012.

Johnathan Vee Cree, Jose Delgado-Frias, Mike Hughes, Brion Burghard, and Kurt Silvers. NOA: A Scalable Multi-Parent Clustering Hierarchy for WSNs. *Procedia CS*, 10:1140–1145, 2012.

S. Du, A. Khan, S. PalChaudhuri, A. Post, A.K. Saha, P. Druschel, D.B. Johnson, and R. Riedi. Self-organizing hierarchical routing for scalable ad hoc networking. *Rice University, Houston, TX, USA, Tech. Rep. TR04-433*, 2004.

S. Du, A. Khan, S. PalChaudhuri, A. Post, A.K. Saha, P. Druschel, D.B. Johnson, and R. Riedi. Safari: A self-organizing, hierarchical architecture for scalable ad hoc networking. *Ad Hoc Networks*, 6(4):485–507, 2008.

Vladimir Dyo, Stephen A. Ellwood, David W. Macdonald, Andrew Markham, Niki Trigoni, Ricklef Wohlers, Cecilia Mascolo, Bence Pásztor, Salvatore Scellato, Kharsim Yousef, et al. WILDSENSING: Design and deployment of a sustainable sensor network for wildlife monitoring. *ACM Trans. Sen. Netw.*, 8(4):29:1–

29:33, September 2012. ISSN 1550-4859. doi: 10.1145/2240116.2240118. URL http://doi.acm.org/10.1145/2240116.2240118.

Amre El-Hoiydi and Jean-Dominique Decotignie. WiseMAC: An Ultra Low Power MAC Protocol for Multi-hop Wireless Sensor Networks. In *ALGOSENSORS*, volume 3121 of *Lecture Notes in Computer Science*, pages 18–31. Springer, 2004. ISBN 3-540-22476-9.

Muhammad Omer Farooq, Abdul Basit Dogar, and Ghalib Asadullah Shah. MR-LEACH: Multi-hop Routing with Low Energy Adaptive Clustering Hierarchy. In *Proceedings of the 2010 Fourth International Conference on Sensor Technologies and Applications*, SENSORCOMM '10, pages 262–268, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4096-2. doi: 10.1109/SENSORCOMM.2010.48. URL http://dx.doi.org/10.1109/SENSORCOMM.2010.48.

E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi. In-network aggregation techniques for wireless sensor networks: a survey. *Wireless Communications, IEEE*, 14(2):70–87, 2007.

Deepak Ganesan, Ben Greenstein, Deborah Estrin, John S. Heidemann, and Ramesh Govindan. Multiresolution storage and search in sensor networks. *TOS*, 1(3):

277–315, 2005. URL http://dblp.uni-trier.de/db/journals/tos/tos1.html#GanesanGEHG05.

Jacob Hagouel. *Issues in routing for large and dynamic networks*. PhD thesis, Columbia University, New York, NY, USA, 1983. AAI8327222.

Matthias Handy, Marc Haase, and Dirk Timmermann. Low energy adaptive clustering hierarchy with deterministic cluster-head selection. In *MWCN*, pages 368–372. IEEE, 2007.

IEEE. IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks Specific Requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). *IEEE Std 802.15.4-2003*, pages 0_1–670, 2003. doi: 10.1109/IEEESTD.2003.94389.

Kwang il Hwang, Jeongsik In, and Doo Seop Eom. Distributed Dynamic Shared Tree for Minimum Energy Data Aggregation of Multiple Mobile Sinks in Wireless Sensor Networks. In Kay Römer, Holger Karl, and Friedemann Mattern, editors, *EWSN*, volume 3868 of *Lecture Notes in Computer Science*, pages 132–147. Springer, 2006. ISBN 3-540-32158-6.

K. Iwanicki and M. Van Steen. Multi-hop cluster hierarchy maintenance in wireless

sensor networks: A case for gossip-based protocols. *Wireless Sensor Networks*, pages 102–117, 2009.

Konrad Iwanicki and Maarten van Steen. Gossip-Based Self-Management of a Recursive Area Hierarchy for Large Wireless SensorNets. *IEEE Trans. Parallel Distrib. Syst.*, 21(4):562–576, 2010.

C. Jardak, E. Osipov, and P. Mahonen. Distributed Information Storage and Collection for WSNs. In *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*, pages 1–10. IEEE, 2007.

M.K. Jha and TP Sharma. Secure Data aggregation in Wireless Sensor Network: A Survey. *International Journal of Engineering Science*, 3, 2011.

Y. Jin, L. Wang, Y. Kim, and X. Yang. EEMC: An energy-efficient multi-level clustering algorithm for large-scale wireless sensor networks. *Computer Networks*, 52(3):542–562, 2008.

N. Kimura and S. Latifi. A survey on data compression in wireless sensor networks. In *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, volume 2, pages 8–13. IEEE, 2005.

V. Kulathumani, A. Arora, M. Demirbas, and M. Sridharan. Trail: A distance sen-

sitive wsn service for distributed object tracking. *Wireless Sensor Networks*, pages 83–100, 2007.

N GNU General Public License. GNU General Public License, version 2, June 1991. In Sam and Stone DiBona, editor, *Open Sources. Voices of the Open Source Revolution*. O'Reilly, Sebastopol, CA, 1999. URL http://www.gnu.org/copyleft/gpl.html.

M. Mamuny, N. Nakaya, Y. Hagihara, G. Chakraborty, et al. HEHC: Heterogeneous-aware enhanced hierarchical clustered scheme for wireless sensor networks. In *SICE Annual Conference (SICE), 2011 Proceedings of*, pages 1517–1522. IEEE, 2011.

Steven McCanne, Sally Floyd, Kevin Fall, Kannan Varadhan, et al. Network simulator ns-2, 1997.

K Muller and Tony Vignaux. SimPy: Simulating Systems in Python. *ONLamp. com Python Devcenter*, 2003.

T. Naumowicz, R. Freeman, H. Kirk, B. Dean, M. Calsyn, A. Liers, A. Braendle, T. Guilford, and J. Schiller. Wireless sensor network for habitat monitoring on Skomer Island. In *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, pages 882–889. IEEE, 2010.

B. Nazir and H. Hasbullah. Energy Efficient Multi Hierarchy Clustering Protocol for Wireless Sensor Network (EMHC). *ISRN Communications and Networking*, 2010.

NS-3. NS-3: Network Simulator 3, 2009.

B.C. Okeke and KL Law. Multi-level clustering architecture and protocol designs for wireless sensor networks. In *Proceedings of the 4th Annual International Conference on Wireless Internet*, page 5. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

L.M. Oliveira and J.J. Rodrigues. Wireless sensor networks: a survey on environmental monitoring. *Journal of communications*, 6(2):143–151, 2011.

Santashil PalChaudhuri, Rajnish Kumar, Richard G. Baraniuk, and David B. Johnson. Design of Adaptive Overlays for Multi-scale Communication in Sensor Networks. In Viktor K. Prasanna, S. Sitharama Iyengar, Paul G. Spirakis, and Matt Welsh, editors, *DCOSS*, volume 3560 of *Lecture Notes in Computer Science*, pages 173–190. Springer, 2005. ISBN 3-540-26422-1.

A. Quiroz and M. Parashar. Design and implementation of a distributed content-based notification broker for ws-notification. In *Grid Computing, 7th IEEE/ACM International Conference on*, pages 207–214. IEEE, 2006.

R. Rajagopalan and P. K. Varshney. Data-aggregation techniques in sensor networks:

a survey. *Commun. Surveys Tuts.*, 8(4):48–63, October 2006. ISSN 1553-877X. doi: 10.1109/COMST.2006.283821. URL http://dx.doi.org/10.1109/COMST.2006.283821.

Thomas Schmid, Roy Shea, Zainul Charbiwala, Jonathan Friedman, Mani B. Srivastava, and Young H. Cho. On the interaction of clocks, power, and synchronization in duty-cycled embedded sensor nodes. *TOSN*, 7(3), 2010. URL http://dblp.uni-trier.de/db/journals/tosn/tosn7.html#SchmidSCFSC10.

Surender Soni and Narottam Chand. Energy Efficient Multi-Level Clustering To Prolong The Lifetime of Wireless Sensor Networks. *CoRR*, abs/1005.4031, 2010.

H.W. Tsai, C.P. Chu, and T.S. Chen. Mobile object tracking in wireless sensor networks. *Computer communications*, 30(8):1811–1825, 2007.

Yu-Chee Tseng, Sheng-Po Kuo, Hung-Wei Lee, and Chi-Fu Huang. Location tracking in a wireless sensor network by mobile agents and its data fusion strategies. In *Information Processing in Sensor Networks*, pages 554–554. Springer, 2003.

András Varga and Rudolf Hornig. An overview of the OMNeT++ simulation environment. In Sándor Molnár, John Heath, Olivier Dalle, and Gabriel A. Wainer, editors, *SimuTools*, page 60. ICST, 2008. ISBN 978-963-9799-20-2.

Yun Wang, Peizhong Shi, Kai Li, and Jie Wu. DQSB: A Reliable Broadcast Pro-

tocol Based on Distributed Quasi-Synchronized Mechanism for Low Duty-Cycled Wireless Sensor Networks, July 2012. URL http://arxiv.org/abs/1207.2604.

Elias Weingärtner, Hendrik vom Lehn, and Klaus Wehrle. A Performance Comparison of Recent Network Simulators. In *ICC*, pages 1–5. IEEE, 2009. URL http://dx.doi.org/10.1109/ICC.2009.5198657.

M. Xie, S. Han, B. Tian, and S. Parvin. Anomaly detection in wireless sensor networks: A survey. *Journal of Network and Computer Applications*, 2011.

Won-Ju Yoon, Sang-Hwa Chung, and Seong-Joon Lee. Implementation and performance evaluation of an active RFID system for fast tag collection. *Computer Communications*, 31(17):4107–4116, 2008. URL http://dblp.uni-trier.de/db/journals/comcom/comcom31.html#YoonCL08.